An Advanced Computational Approach to System of Systems Analysis & Architecting Using Agent-Based Behavioral Model

Final Technical Report SERC-2013-TR-021-3

November 18, 2013

Principal Investigator: Dr. Cihan H. Dagli, Professor and Systems Engineering Program Director, Missouri University of Science & Technology

Team Members

Dr. Nil Ergin, Assistant Professor - Pennsylvania State University

Dr.David Enke, Department Chair and Professor - Missouri University of Science & Technology

Dr. Kristin Giammarco, Associate Professor - Naval Postgraduate School

Dr. Abhijit Gosavi, Assistant Professor - Missouri University of Science & Technology

Dr. Ruwen Qin, Assistant Professor - Missouri University of Science & Technology

Dr. Dincer Konur, Assistant Professor - Missouri University of Science & Technology

Dr. John Colombi, Assistant Professor - Air Force Institute of Technology

Siddhartha Agarwal, Khaled Haris, Louis Pape, PhD Students - Missouri University of Science & Technology

Contract Number: H98230-08-D-0171

WHS TO 029, RT 044

Report No. SERC-2013-TR-021-3
November 18, 2013

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
18 NOV 2013	**Final**	

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
An Advanced Computational Approach to System of Systems Analysis & Architecting Using Agent-Based Behavioral Model, Phase - 2	**H98230-08-D-0171**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)	5d. PROJECT NUMBER
Dagli /Dr. Cihan	**RT 44-6**
	5e. TASK NUMBER **WHS TO 029**
	5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Stevens Institute of Technology Missouri University of Science & Technology Pennsylvania State University Missouri University of Science & Technology Naval Postgraduate School Air Force Institute of Technology	**SERC-2013-TR-021-3**

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
DASD (SE)	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release, distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The goal of this research is to model the evolution of the architecture of an acknowledged Systems of Systems (SoS) that accounts for the ability and willingness of constituent systems to support the SoS capability development. Since DoD SoS development efforts do not typically follow the program of record acquisition process described in DoDI 5000.02, the Wave Model proposed by Dahmann and Rebovich is used as the basis for this research on SoS capability evolution. The Wave Process Model provides a framework for an agent-based modeling methodology, which is used to abstract the non-utopian behavioral aspects of the constituent systems and their interactions with the SoS. In particular, the research focuses on the impact of individual system behavior on the SoS capability and architecture evolution processes.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**UU**	**158**	

This material is based upon work supported, in whole or in part, by the U.S. Department of Defense through the Systems Engineering Research Center (SERC) under Contract H98230-08-D-0171. SERC is a federally funded University Affiliated Research Center managed by Stevens Institute of Technology

Executive Summary

A major challenge to the successful planning and evolution of an Acknowledged System of Systems (SoS) is the current lack of understanding of the impact that the presence or absence of a set of constituent systems has on the overall SoS capability. Since the candidate elements of a SoS are fully functioning, stand-alone systems in their own right; they have goals and objectives of their own to satisfy, some of which may compete with those of the overarching SoS. These system-level concerns drive decisions to participate (or not) in the SoS. Typically, individual systems are invited to join the SoS construct, and persuaded to interface and cooperate with other Systems to create the "new" capabilities of the proposed SoS. Current SoS evolution strategies lack a means for modeling the impact of decisions concerning participation or non-participation of any given set of systems on the overall capability of the SoS construct. Without this capability, it is difficult to optimize the SoS design.

The goal of this research is to model the evolution of the architecture of an acknowledged SoS that accounts for the ability and willingness of constituent systems to support the SoS capability development. In particular, the research focuses on the impact of individual system behavior on the SoS capability and architecture evolution processes.

The agent based model (ABM) structure is developed to provide an Acknowledged SoS manager a decision making tool in negotiation of SoS architectures during wave model cycles The overall ABM consists of 3 major elements; SoS acquisition environment, SoS agent, and a system agent. Each agent has its own set of behavior patterns. SoS meta- architecture obtained from one of the SoS meta-architectures generation modules, namely; Fuzzy –genetic optimization model, Multi-Level Optimization model, and Multi-objective optimization model, drives the negotiation process. ABM also provides alternatives for participating systems to choose from three types of negotiation models. The negotiation model for SoS is fixed.

The ABM has one instance of the SoS agent and multiple instances of the system agent. The number of instances of the system agent corresponds to the number of systems in the SoS. This approach helps create multiple alternatives to generate architectures for acknowledged SoS. An Intelligence, Surveillance and Reconnaissance (ISR) SoS, consisting of 22 individual systems with five capabilities, is used as a domain example to demonstrate the framework of the ABM for one wave cycle.

The current analysis environment has matured to the point where it could support SoS analysis and decision-making, which would identify new opportunities to improve the SoS analysis tools. The next step is to create the demonstration and presentation materials necessary to describe the capabilities of the ABM, and provide an overview of analysis tools to potential users identified by the sponsor.

Contents

List of Figures

List of Tables

1 Introduction

The goal of this research is to model the evolution of the architecture of an acknowledged Systems of Systems (SoS) that accounts for the ability and willingness of constituent systems to support the SoS capability development. Since DoD SoS development efforts do not typically follow the program of record acquisition process described in DoDI 5000.02, the Wave Model proposed by Dahmann and Rebovich is used as the basis for this research on SoS capability evolution. The Wave Process Model provides a framework for an agent-based modeling methodology, which is used to abstract the non-utopian behavioral aspects of the constituent systems and their interactions with the SoS. In particular, the research focuses on the impact of individual system behavior on the SoS capability and architecture evolution processes.

1.1 Motivation

This research develops, validates, and pilots ABM Methods, Tools and Processes (MTPs) that support investigating the properties of an acknowledged SoS, and can be applied early in the life cycle when there is uncertainty and ambiguity about SoS requirements, architecture, DoD Acquisition guidance and implementation technologies, based on the Wave Process Model.

A generic ABM framework provides a capability to model the entire SoS in the context of its environment, which is distinct from a more typical modeling of one system in the context of its environment. There are many moving pieces in such a model, and that complexity introduces many opportunities to select suboptimal designs for an Acknowledged SoS. We cannot afford to continue to make decisions "eyeballing" the SoS design and improving through trial and error. We need to use modern tools to model our understanding of the behavior of the SoS under difference circumstances (such as participation/non-participation of systems) so that we have clear visibility into the full range of feasible SoS designs as well as tradeoffs among those designs, and see the predictable impacts in advance of decisions. No structured, repeatable approach is consistently used within DoD for planning and modeling proposed alternative Acknowledged SoS architectures, and then systematically prioritizing those architectures in order of the best to worst match with stakeholder needs like flexibility, robustness, performance, etc. This research takes a step towards achieving that capability by introducing a new analysis framework that uses modern modeling tools to expose foreseeable SoS level impacts for decision makers early in the lifecycle, when such impacts can be managed less expensively and more solutions to possible problems can be put on the table.

Early insight into likely properties of acknowledged SoS meta architectures and conditions that affect their implementation and effectiveness will inform technological and policy decisions about far reaching and expensive SoS acquisition decisions.

1.2 Research Objectives

The goal of this research is to develop a proof of concept ABM tool suite for SoS systems simulation for architecture selection and evolution. An Intelligence, Surveillance and Reconnaissance (ISR) SoS, consisting of 22 individual systems with five capabilities, is used as a domain example to demonstrate the framework of the ABM tool suite. A second example, using an ad hoc SoS in the Search and Rescue domain, shows that the framework is extensible to other domains of application. The most far-reaching objective is demonstrating the interoperability among the framework and several optimization models and several agent models combined here.

Policies on architecting in the DoD continue to evolve, although perhaps at a slower pace than in the recent past. Partly due to this evolution, but also because analysis techniques of very large alternative architectures is in its infancy, the modeling of architecture development and evolution is not settled science. This is particularly true in SoS settings [1]. Existing analysis methodologies and tools narrow the scope of the SoS problem space by invoking the assumption that there is a limited set of solutions, solely or primarily driven by technical performance considerations. However, the SoS problem boundary includes integration of technical systems as well as cognitive and social processes, which alter system behavior [2].

As mentioned before, most system architects assume that SoS participants exhibit nominal (utopian) behavior, but deviation from nominal motivation leads to complications and disturbances in desired systems' behavior. It is necessary to capture the behavioral dimension of SoS architecture to be able to represent the full problem space to guide the SoS architecting and analysis phase [2]. Evaluation of architectures lends itself to a fuzzy approach because the criteria are frequently non-quantitative, subjective, or based on unknowable future conditions (such as "robustness"). Finally, since one of the current problems with SoS composition is lack of imagination, and system participation or non-participation is conveniently binary, a genetic algorithm (GA) as one of the non-gradient based optimization approaches can help to explore a wider potential architecture space more fully.

Agent based models (ABMs) consist of a set abstracted entities referred to as agents, and a framework for simulating agent decisions and interactions. Agents may have their own goals and are capable of perceiving changes in the environment [3]. SoS behavior (global behavior) emerges from the decisions and interactions of the agents acting as individuals. This approach may provide insight into complex, interdependent processes. Agent-based modeling methodology has several benefits over other modeling techniques; it enables Acknowledged SoS manger to capture emergent patterns of system behavior, provides a natural description of a system composed of behavioral entities and is flexible for tuning the complexity of the entities [4]. The methodology is used in a wide range of application domains including financial markets [5], homeland security applications [6] and autonomous robots [7].

Agent-based modeling methodology is used to abstract behavioral aspects of the acquisition process. In this project, the systems are assumed as the agents. The System Agents embody themselves and the people (individual stakeholders) responsible for them. The wave model applies to an Acknowledged SoS, thus there is also a specific agent responsible for the SoS, and that agent influences the other

System Agents. An initial SoS mission is initially determined and funds are allocated to the mission from a responsible organizational entity [8]. The structure of the wave model is depicted in Figure 1 [9].

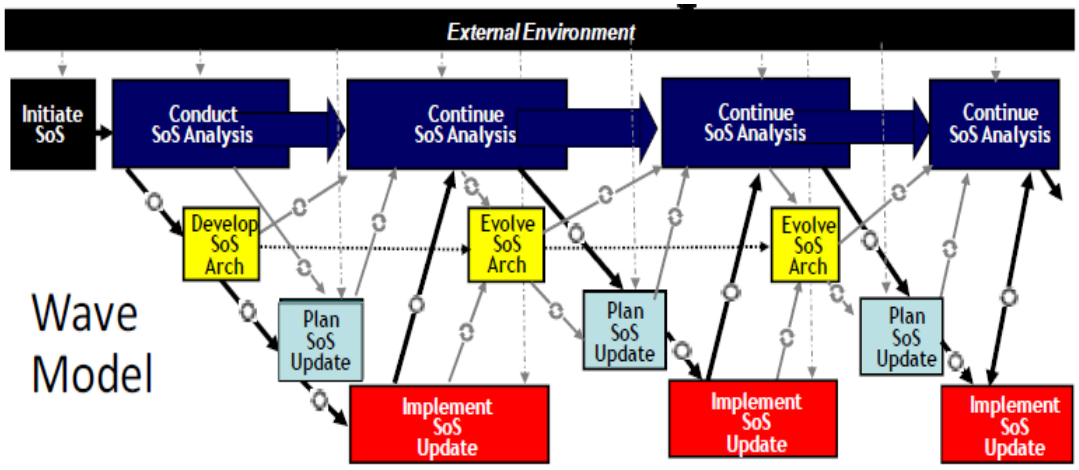

Figure 1. The Wave Model of SoS initiation, engineering, and evolution

2 Background and Literature Review

2.1 Systems of Systems Challenges

System of systems (SoS) is a specific case of complex adaptive systems (CAS). To understand the behavior of SoS we must understand not only the behavior of the parts but how they act together to form the behavior of the whole. Tools and models for SoS engineering need to address central properties of CAS including its elements (and their number), interactions (and their strength), formation/operation (and their time scales), diversity/variability, environment (and its demands), activities and their objectives.

Like any CAS, System of systems is at the edge of chaos. It tries to maintain dynamic stability through constant self-adjustment and evolution. As shown in Figure 2, chaos and order are two complementary states of our world. A dynamic balance exists between these two states. Order and structure are vital to life. Order ensures consistency and predictability and makes the creation of systems possible. However, too much order leads to rigidity and suppresses creativity. Chaos constantly changes the environment creating disorder and instability but can also lead to emergent behavior and allows novelty and creativity. Thus, sufficient order is necessary for a system to maintain an ongoing identity, along with enough chaos to ensure growth and development. The challenge in SoS engineering is to design an organized complexity that will allow SoS to achieve its goals. Designing a balanced and organized complexity is not a trivial task because SoS capabilities can conflict with constituent system requirements and plans. Furthermore, individual systems can belong to more than one SoS where conflicts can arise due to different change requests to the constituent systems. The need for techniques

to enable reductions in SoS complexity and enable managed evolution has been pointed in the literature.

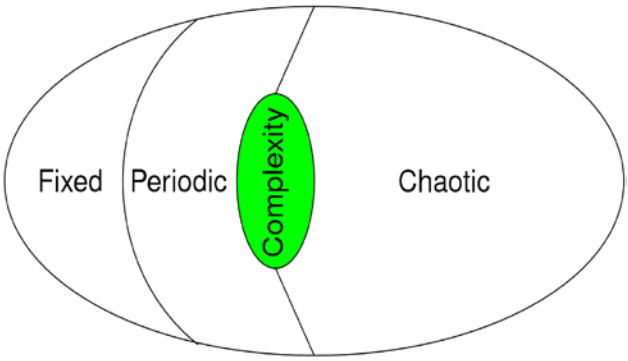

Figure 2. The edge of chaos, the point of emergence and SoS

Complexity methods analyze systems and its parts in the context of the whole while considering its interaction with its environment as the environmental influence is important in describing the behavior of the system. Modeling and simulation used in the study of CAS to understand emergent behaviors of systems, discover its dynamic behavior, and identify intra and interdependencies among its elements and environment. Tools for modeling and studying complex systems include nonlinear dynamics, agent-based models and network theory based models where system represented as a network consisting of vertices (constituent systems) and edges (interactions).

In the SoS domain, modeling and simulation becomes an important tool for exploring new capability options, identifying integration problems, evaluating SoS architecture and supporting SoS level testing [10]. Recent studies of SoS offer various modeling approaches for SoS engineering. For example, Lane and Bohn [10] present an approach to quickly generate models using SysML models in order to support decision making in SoS. Haimes [11] extends hierarchical holographic modeling (HHM) to phantom system models (PSM). HHM captures multiple features of a system that may reside in different levels of a system. This modeling approach is used to study risks for government agencies. PSM builds on to HHM by providing operational guidelines and principles to model a meta-model for SoS. Meta-model of the SoS serves as the coordinator and integrator of the multiple models; building on the shared and unshared state variables. Garett et al [12] present a framework for integrating complex systems into SoS functional capability. Graph theory along with agent-based modeling and other simulation models are explored in their framework to address interfaces, interoperability and integration aspects of SoS engineering.

Architecture modeling and evaluation plays a key role in SoS engineering for exploring logical, behavioral and performance characteristics of SoS in order to acquire desired capabilities. Static architectural frameworks are more suited for traditional systems and current executable models construct models by manually transforming architectural descriptions with weak model consistency checking [13]. A data-centric SoS architecture modeling approach is proposed in Ge et a [13] for iterative architecture refinement where SoS requirements are changing. The approach constructs a high-level data meta-

model to describe semantic relationships among constituent system architectural data elements. This meta-data enables construction of executable models based on core data elements while preserving semantic consistency and traceability in architectural models.

Evaluation of architectures is another SoS challenge area as it lends itself to a fuzzy approach because the criteria are frequently non-quantitative, or subjective [14], or based on difficult to define or even unknowable future conditions, such as "robustness." Individual attributes may not have a clearly defined, mathematically precise, linear functional form from worst to best. The goodness of one attribute may or may not offset the badness of another attribute. Several moderately good attributes coupled with one very poor attribute may be better than architecture with all marginally good attributes, or the reverse! A fuzzy approach allows many of these considerations to be handled using a reasonably simple set of rules, as well as having the ability to include non-linear characteristics in the fitness measure. The simple rule set allows small adjustments to be made to the model to see how seemingly small changes affect the outcome.

There are other SoS engineering research initiatives including the European Commission's 7th Framework Programme, which is currently funding four key projects. T-AREA-SoS is an EC project, dedicated to develop a strategic research agenda in SoS engineering that is mutual to the EU and USA. Initial set of themes identified for strategic research agenda include [15]:

- Theoretical foundations of SoS,
- Characterization and description of SoS,
- Predicting and management of emergence,
- Measurement and metrics for SoS,
- Multi-level modeling,
- Evaluation of SoS,
- Human and organizational aspect within SoS,
- Trade-off at the SoS level,
- Prototyping of SoS,
- Definition and evolution of SoS architecture,
- Energy efficiency in SoS,
- Cyber-security aspects in SoS

RoadtoSoS, another EC IP project dedicated to SoS, focuses on research, technology, development, and innovation strategies for Europe in four major SoS engineering applications: distributed energy generation and smart grids, integrated multi-site industrial production, emergency and crisis management, and multi-model traffic control. DANSE project focuses on developing a tool set to specify SoS and SoS requirements at high level using Unified Profile for DoDAF and MoDAF (UPDM), and analyze them at lower levels using simulation toolsets and statistical model checking for formal verification [16]. The goal is to apply this toolset to dynamically evolve SoS architecture and optimize SoS architecture, design and validate through simulation tools. The toolset is currently being applied to various

application areas including development of dynamic water treatment SoS architecture, development of dynamic Air Traffic Management SoS, and development of autonomous ground transport SoS. The DANSE project focuses on formal semantics for modeling SoS language to facilitate analysis and dynamical reconfiguration. Using this formal semantics, a toolset is developed by integrating various SoS analysis and simulation tools with commercial design tools. COMPASS, another EU IO project dedicated to SoS, focuses on developing a formal notation called COMPASS Modeling Language (CML) which extends SysML, VDM and Circus notations for SoS modeling and analysis. The CML is intended to be accessible to a wide range of developers building different SoS models and provides different levels of description including a graphical view for stakeholders to understand. SoS architectures and contracts can be expressed using the CML formal modeling language [17]. The tool is applied in two case studies; Video/Audio/Home automation ecosystem, and Accident Response System.

It is envisioned that integrating various modeling approaches that address different aspects of the SoS engineering problem provides insights and systems thinking about various dimensions of the SoS engineering problem. The methodology outlined in this research and technical report falls under a multi-level plug-and-play type of modeling approach to address various aspects of SoS acquisition environment: SoS architecture evaluation, SoS architecture evolution, and SoS acquisition process dynamics including behavioral aspects of constituent systems.

2.2 Acknowledged SoS

It has been previously recognized that the architecture for an Acknowledged SoS is an overlay on existing systems that have their own architectures [18]. Moreover, successful Acknowledged SoS managers understand that existing system architectures work best if left to the systems engineering and architecture professionals at that hierarchy. It is in the best interest of the SoS domain manager to co-ordinate and guide individual systems rather than foist his own ideas for success. On the contrary, the constituent-systems do not need to acquiesce to SoSE requests or officially report to SoSE teams.

This modeling framework presented is for an Acknowledged System of Systems (SoS), where each component system is a fully functioning, independently funded and managed System (or Program Office). A high-level manager foresees the opportunity to achieve a needed, new capability by using existing systems in a way such that they are kept unchanged or incorporated with relatively minor system changes. The SoS approach is only useful if it can achieve the new capability for either or both the reduced cost compared to designing a separate, new “purpose built” system, and/or reduced time to field the new capability. The architecture proposed is a novel architecture, which will be called the meta-architecture throughout the report. It will guide the SoS system generation through the wave model.

The new capabilities being sought from the SoS are achieved by combining mostly *existing* System capabilities and/or adding new capabilities that arise *in conjunction* with *other* Systems (i.e., through new interfaces) [19]. If simply throwing more Systems (with their individual capabilities) at the problem sufficed, there would be no need to create the SoS. Therefore, all successful Acknowledged SoS architectures need to spend significantly on the relationships between the systems and the SoS. The

nature of the *Acknowledged* SoS though, means that the SoS manager does not have absolute authority to command participation, but must "purchase" the component Systems' participation and modifications not merely with funds but also through persuasion, the strength of the vision of the SoS, quid pro quos, the bully pulpit and whatever other means are legitimate and effective [8]. Individual Systems remain free to decide *not* to participate in the SoS development. Alternatively, they may not be available during a particular operational period of need. Some capabilities and interfaces already exist, meaning they are free and fast for development, but they may still have a cost to operate in the fielded SoS. Some systems may have enough capability that the SoS can tap their spare capability, so they are essentially free to operate as well. Other capabilities may need minor (compared to a major program) development, either within a system, or a new interface with another system. The Performance capabilities of the SoS will generally be greater than the sum of the capabilities of its parts [20]. If this were not the case, there would be no need for the SoS. Changing the way the systems interact without modification typically would not improve the capabilities by simply adding more systems that are individual. Systems engineering in the overall context SoS System Engineering (SE) must address from first to last all behavior of joint group of systems, and the crucial issues, which affect that behavior. An instance of Acknowledged SoS is a military command and control SoS that has transitioned from a collaborative SoS to an acknowledged SoS due to the importance of the missions supported by the SoS or the complexities of the cross-cutting SoS capabilities [21].

There have been few attempts to describe the architecting method of *Acknowledged* SoS. One such approach is based on the federated architecture (FA) [22]. FA is a pattern that describes the construction of the meta-architecture. This approach emphasizes to allow interoperability and information sharing between constituent systems and the centralized controller. Another approach has been to model the interdependencies of systems and impacts of failures using Bayesian networks. The outcomes of the Bayesian analysis with failure rates modeled as beta distributions provide a knowledge base for decision makers to control risk in development of a SoS with complex interdependencies. [23]

The SoS acquisition environment is affected by several external factors such as changes in the national priorities, changes in the SoS funding and changes in threats to the nation. Please refer to Figure 3. Thus, the initial environment model E_0 can be represented as a function of these variables.

$$E_T = f(National\ priorities, SoS\ funding, threats)$$

The SoS agent is influenced by the changes in the SoS acquisition environment. Let E_T be the initial environment model which represents the SoS acquisition environment at wave time T=0.

As the SoS acquisition progresses through wave cycles, these variables are updated by the SoS Acquisition Manager to reflect current acquisition environment. Thus the environment model E_{T+1} at wave time T+1:

$$E_{T+1} = E_T \sigma_{T+1}$$

where σ_{T+1} is the change from time T in external factors at wave time T+1.

The environment model also includes two types of agents, the SoS agent and the individual system agent. The SoS agent is responsible for the development of the SoS architecture and the instances of the individual system agent represent independent systems that can provide the capabilities necessary for the SoS agent. One cycle through the *proposal-negotiation-agreement-execution* development steps is an epoch in the wave model [21]. Once the SoS and participating systems agree on a SoS architecture it culminates in a wave of the modeling cycle.

The SoS architecture after the end of the wave is saved for as a chromosome and serves as a starting point for the start of the next wave.

Within the epoch cycles, the Agent Based Model lets the SoS Agent make cooperation requests to the individual Systems' independent Agents, which have relatively simple rules for their negotiation strategies. We currently have three types of system agent negotiation strategies: selfish, opportunistic-Markov, and cooperative. The ABM portion of the framework manages the negotiations, and returns an "achievable" systems negotiated architecture/chromosome for re-evaluation by the fuzzy inference system. SoS Agent uses this negotiated architecture to implement and making it an initial condition and input for the following wave meta-architecture optimization models.

Some prominent characteristics of Acknowledged SoS are:

1. A strong organizational relationship between the Acknowledged SoS manager and participating systems is important to address constraint goals over the life of the SoS.
2. Acknowledged SoS should be pliable enough to accommodate any variations in individual systems and reduce the impact on the constituent systems considerably.
3. An acknowledged SoS should try to fall between feasible and trivial architectures.
4. Overall architecture quality of Acknowledged SoS fuzzy because it includes many qualitative attributes besides quantitative ones. These qualitative attributes cannot be easily measured or tested.

These characteristics provide the impetus to develop a methodology of architecting Acknowledged SoS using genetic algorithms and fuzzy inference systems. Genetic algorithms help in representing the systems and their interfaces whereas fuzzy logic helps in evaluating the overall architecture quality by qualitative attributes. The domain model (comprising systems and their relationships) is described in terms of bits in the chromosome to evaluate cost, capability contribution, and time to deliver. The fuzzy assessor evaluates these chromosomes for fitness in terms of overall architecture quality. The GA further manages the exploration of the architecture space and picks an optimum or ideal solution.

The model provides a set of systems and their interfaces with other systems as a linear array of bits; zero indicates non-participation in a performance capability, and one represents participation. If a

System does not participate, then no interface with that System is possible. Incorporating a fuzzy evaluation in the process allows substantial nonlinearity to be introduced, but controlled by a simple, short list of rules.

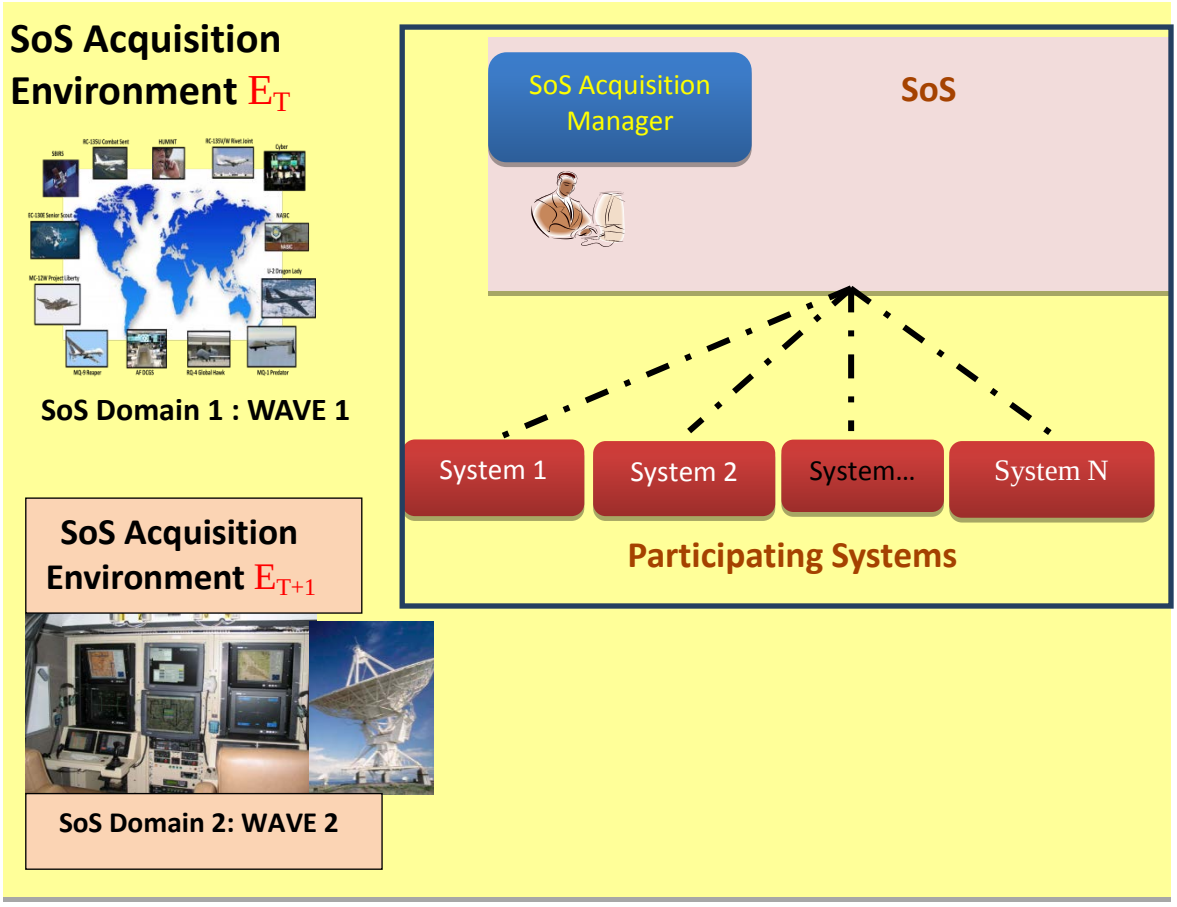

Figure 3. Acknowledged SoS acquisition described as a wave model

2.3 Representing and Assessing SoS Architecture Information

One of the problems with SoS is that they frequently cross domains; they either address broad new problem areas that are not traditionally understood as having connections, or develop because of changes in technology. Either way, analyzing this type of problem requires extensions to the old ways of thinking about or doing things. Simply describing the characteristics, boundaries, expectations, or governance of a SoS is difficult, being fraught with no commonly accepted terms for the new area, little agreement on what constitutes success, nor even a good theory of SoS [24]. The acknowledged SoS that is the focus of this effort only acerbates these problems because of the inherent limits of the responsibility, authority and accountability between the SoS manager and the system program offices who participate in the SoS formation [8] [25]. The literature describing SoS engineering is growing in coverage, but it is still sparse. Pitsko and Verma [25] describe four principles to make a SoS more

adaptable. They spend a large part of their time describing what adaptable means to various stakeholders, that different stakeholders may continue to have slightly different concepts of what adaptability means, that the definition is probably dynamic – changing over time, and that this will hold for many SoS attributes. Schreiner and Wirthlin discuss a partial failure to fully model a space detection SoS architecture, but learned a lot about how to improve the next time they try it [26]. The point is that people are not modeling ahead of time according to a theory then reporting on the success.

The differences between SoS engineering and systems engineering are discussed by Flanagan and Brouse [27], pointing out that different sorts of trade spaces open up in SoS. Some of the concepts about flexibility in this report trace to the options and limiting risk in DoD programs discussion from Giachetti [28]. Countering some of these difficulties in describing SoS architectures are the advances in describing complex systems with fuzzy sets. There are numerous approaches in the literature attempting to describe useful attributes that can be measured to help understand or predict the value of various architectures. These include evolvability and modularity almost as complementary attributes [29], while Christian breaks evolvability into four components described as extensibility, adaptability, scalability and generality [30]. Christian introduces the concept of complexity to overlay on these attributes because too simple a system cannot evolve. Kinnunen reviews at least four definitions of complexity [31] before offering his analysis of one related to the object process methodology of Dori, and Mordecai and Dori extend that to SoS specifically for interoperability [32]. Fry and DeLaurentis also discussed measuring Net-Centricity (interoperability within the SoS), noting the difficulty of pushing the commonly used heuristics too far, because the Pareto front exists in multiple dimensions [33], not just two at a time. Ricci et al. discuss designing for evolvability of their SoS in a wave model and playing it out several cycles in the future, evaluating cost and performance. [34] Because SoS are complex, there are many ways to look at them, with no dominant theory yet; this is why we are pursuing this direction of research [35].

We selected slightly different definitions for some of our SoS attributes, especially for flexibility and robustness. Lafleur used flexibility in the operational context of changing after deployment [36], in a way that sort of trumped Deb's notion of robustness [37] by shifting the optimum (narrowly better performance), rather than accepting lower performance across a wider front; we preferred to use robustnesss in the operational context suggested by Singer [38] of losing a node in a network, rather more like our losing a system from our SoS. Our concept of the flexibility attribute is more attuned to giving the SoS manager flexibility during development, while picking systems to supply desired capabilities. This falls in line with some of the thinking of recent discussions of resilience and sensitivity analyses [39] [40] [41].

Mendel notes that there are numerous fuzzy approaches allowing, "Computing with words" and extracting meaning even from the degree of our lack of knowledge to be included in solving a large variety of problems [42]. Li and Chiang [43] introduce the concept of complex fuzzy sets, which even replace the *if-then* rules of Mamdami fuzzy systems. Many of these techniques are more applicable to extremely large data sets such as those of social media where polling a huge population can detect trends and shifts in public opinion on the time scale of hours, rather than subject matter expert opinions

on engineering tasks. We primarily used simpler, older more straightforward techniques for this first modeling approach.

2.4 Negotiation Models for SoS

During an interaction between entities/agents/systems that are trying to solve a problem two kinds of altitudes can occur: Cooperation if all systems have the same goal, negotiations if self-interests are in conflict. In SoS where individual systems have their own self-interests, negotiation becomes an important aspect of SoS acquisition. During a negotiation, each party communicates its own desires and then compromises to reach a mutually beneficial agreement. In negotiation, each party would like to reach an agreement rather than disagreement but the agreement should be as favorable to each party as possible.

In real life negotiation situations there are three main features of a negotiation process (Fatima et al, 2004):

- Time constraints of the negotiators
- Information state of the negotiators
- The number of issues to be bargained over

There are two main approaches to negotiation studies; formal theory of bargaining and informal theories. Formal models use a game-theoretic approach to analyze different situations and results with the aim off finding the best rational strategy for the negotiator. This approach assumes that people involved in the negotiation process are rational and follow certain negotiation protocols. Formal game theoretic models minimize communication among participants but find a solution that maximizes the social welfare of the negotiation participants. Informal approaches (also referred as knowledge based protocols) focus on identifying beneficial strategies for a negotiator and do not impose rationality restrictions. These models depend on trade-offs made by the participants to determine the agreement solution. However, from a computational modeling perspective, informal approaches are more difficult to apply as they do not use formal theories or strategies [44] and they do not guarantee maximization of social welfare for negotiators [45].

In a given negotiation scenario, game theoretic approaches look into two main problems [46]:

- The design of an appropriate protocol to manage the interactions between the negotiation participants. The protocol establishes the rules of engagement among the participants. The design of protocol have certain desirable properties including; guaranteed success for mutual agreement, maximizing social welfare of negotiation participants, Pareto efficiency where there is no outcome that will make one agent better off without making at least one other agent worse off, individual rationality, stability, simplicity and distribution.
- The design of a strategy for the negotiation participants that will maximize their welfare.

Negotiation dynamics have been explored widely in multi-agent negotiation applications. Kraus [44] proposed a strategic-negotiation model that, which is based on game-theoretic approaches and

demonstrated the use of this model in various applications including data allocation problems in information servers, resource allocation and task distribution, and pollution allocation problem. The goal of this approach is to minimize the delays and time spent on reaching mutually beneficial agreements. Fatima et al. [47] proposed a model for multi-issue negotiation under time constraints in an incomplete information setting. They determined the properties of a Pareto-optimal equilibrium solution. Wanyama and Far [45] proposed a negotiation protocol for group choice decision-making problems. Jonker et al [48] developed a negotiation model for multi-attribute negotiation where each negotiator has incomplete preference information about each other. Krothapalli and Deshmukh [49] proposed negotiation protocols for multi-agent manufacturing systems.

In the SoS domain, the study of negotiation dynamics is in its infancy. A fuzzy decision analysis model is developed to represent the negotiation between SoS manager and constituent systems in an acknowledge SoS acquisition environment [50]. Contracting is a special form of negotiation where an organization tries to contract tasks that it cannot perform by itself or where others can perform it more effectively than itself. The challenge in these settings is to convince another organization that is self-interested to agree on collaboration. Incentive contracting is studied in economics and game theory and applied in multi-agent negotiation settings as well [51]. Contracting processes and structures for SoS acquisition are studied in [51] to identify effective contracting structures to maximize the probability of SoS acquisition success.

2.5 Generating Optimum SoS Meta-Architectures

2.5.1 Introduction

While systems architecting involves the process of structuring system components and their interconnections, system of systems (SoS) architecting involves multiple inter-connected systems and can include socio-technical and dynamic dimensions. The concept of meta-systems and meta-architecture's has not been fully crystallized and definition varies among researchers [52]. The concept used here is inspired from the definition provided by [53]. In theory, meta- architecture trades can be generated through mathematical programming models, heuristics and domain dependent algorithms.

This part of the report instantiates SoS meta-architecture generating methodologies. Meta-architecture generation can have various constraints, a large number of objective functions, combinatorial characteristics, and deal with stakeholder preferences to name a few aspects. This modeling method uses three different techniques, namely fuzzy-genetic approach, multi-level optimization and multi-objective optimization, to generate Acknowledged SoS meta-architectures. All these techniques involve temporal exploration of trade space in a changing environment based on concepts of waves and epochs.

2.5.2 Multi-Level Optimization

In many practical applications, multiple agents interact with each other to determine their individual strategies as one's strategy affects the others' outcome. To model the decision making process when multiple agents interact, one needs to consider the decision making sequence among the agents. Decision making sequence can fall into two categories: simultaneous decision-making, i.e., when all of

the agents determine their strategies at the same time and hierarchical decision-making, i.e., there exists an order for the agents to announce their decisions. In the former category, game theoretic approaches are commonly used to model the decision-making processes. In the latter case, the inherent hierarchical decision making process can be modeled using multi-level optimization approach.

In particular, multi-level optimization models are observed when a set of agents has authority to affect the decision making process of the other agents [54]. For the problem of interest in this study, the SoS architect has the authority to negotiate with individual system providers to improve the performance measures of the requested capabilities by means of allocating additional funds. Depending on the funds allocated, the system providers then decide on which performance measures to improve on which of the requested capabilities. That is, there exists an order of decision making between the SoS architect and the system providers: SoS architect is the first decision maker and the system providers are the followers. This order of decision-making is generally referred to as Stackelberg game and it can be formulated as a multi-level optimization model.

In multi-level optimization problems, there exist different optimization problems at different levels, corresponding to the decision making of the agents at different hierarchical order [55]. The optimization problem of an agent with lower hierarchical order is actually included as a constraint within the optimization problem of an agent with higher hierarchical order. Therefore, the agent with the higher hierarchical order can regard the next decision makers' optimum strategies while determining his/her own strategy. For instance, for the SoS architecting problem of this study, the SoS architect should consider how much improvement will be observed in which capabilities requested while determining the funds allocated to system providers. In doing so, SoS architect, therefore, needs to estimate how each system provider will spend the funds allocated.

Multi-level optimization problems are considered to be one of the most challenging classes of optimization problems. Even the single-objective multi-level optimization problems in the linear case are discussed to be NP-hard [56]. The SoS architecting is itself a multi-objective optimization problem and considering the possibility of contracting with individual system providers results in a multi-objective multi-level optimization problem. It is noted that analysis of multi-objective multi-level models is rather limited in the literature [57] [58]. Due to the complexity of such problems, theoretical properties of such problems are limited [see, e.g., [59] and generally heuristics methods are adopted to provide good quality solutions.

Evolutionary methods are one class of heuristic solution approaches adopted for multi-level multi-objective optimization problems. Deb and Sinha [57] study a hybrid evolutionary heuristic method for solving bi-level multi-objective optimization problems. Li et al. [60] use a non-dominated sorting algorithm with genetic algorithms to solve a class of bi-objective bi-level optimization problems. Similarly, Jia et al. [61] propose a variation of genetic algorithms for solving convex multi-objective bi-level optimization models. Konur and Golias [62] construct a met heuristic method including local search and genetic operations to solve a mixed-integer non-linear bi-objective bi-level optimization

problem. Particle swarm optimization method is adopted by Zhang et al. [63] [64] to solve multi-objective bi-level optimization problems.

Fuzzy modeling approaches are also used to solve multi-objective multi-level optimization problems. Osman et al. [65] develop a fuzzy min-max decision model to generate Pareto optimal solutions for a multi-level non-linear multi-objective decision making problem. Lu et al. [66] and Zhang and Lu [67] approximate K-th best approach of a multi-objective multi-level model by using fuzzy techniques to resolve the mutli-level complexity of the model. Gao et al. [68] provide a decision support tool for a fuzzy multi-objective multi-level optimization problem. Finally, Baky [69], [70] uses a fuzzy goal programming method and Zheng et al. [71] use a fuzzy interactive method to solve a class of multi-objective multi-level optimization problems.

Other than the aforementioned approaches to solve multi-objective multi-level optimization problems, there are studies that consider integer programming and reformulation approaches [see, e.g., [59], [58], [72], branch-and-bound type of algorithms (see, e.g., [73], [74]) and interactive procedures [see, e.g., [75], [76]) for such problems. The problem of interest in this study, i.e., SoS architecting with individual system contracts, is formulated as a multi-objective bi-level problem, where the upper level is a mixed-integer nonlinear programming model with three objectives and the lower level consists of an arbitrary number of linear programming models. Considering the complexity the resulting model, an evolutionary meta-heuristic a two-stage solution approach is proposed. The details of the problem formulation and solution approach are explained in Section 3.4.1.

2.5.3 Fuzzy Genetic Approach

Another common approach to the multi-objective optimization problem is to use a genetic algorithm approach with a fuzzy fitness assessor as the chromosome sorter between generations. The combination of multi-objective optimization with fuzzy approaches is discussed by Cara et al. [77]. Their problem was to fit surfaces with minimum error and minimize fuzzy rules while comparing Type-1 vs. Type-2 fuzzy sets, but several of their ideas are incorporated here, such as minimizing the rules in the fuzzy rule base to make architecting the SoS easier to explain to stakeholders. They also showed that Type 1 fuzzy systems are better in low noise (except for the input itself) situations, and Type-2 works better where there is noise in the rest of the system. This effort used the simpler Type 1 fuzzy systems, but an obvious extension using noisier, real world stakeholder linguistic inputs is possible. Wang and Zhang discuss incomplete information and weighted sets, but also include the concept of the penalty function as a more subtle method to push the fuzzy set solution off unwanted or infeasible solutions [78]. Sanz et al. [79] present the method used here of tuning the membership functions and rules to fit the data as only the first part of their paper.

This project attempted to simplify and modularize the treatments of the description, definition of what is important to the stakeholders, SoS attribute selection, development and funding within each system, interactions between member systems when the SoS is fielded, and the negotiation between the SoS manager and the systems. A major effort was the segmentation and re-integration of the models so that a variety of techniques could be tested with each other. The modularity was desired because it was

not known what techniques would work best together, nor if different types of problems would require different approaches. There are a large number of genetic algorithm techniques [80], [81], from the very simple constant mutation rate on all the population, to sexual crossover, to variable size but gene specific transpositions. In selecting chromosomes for reproduction to the next generation, techniques range from simple tournament selection of the best few, to roulette based 'higher fitness gives more chance of random selection (but not a guarantee)' for reproduction. The big driver is the choice of representation of the problem, the size of the domain, whether the gene components of the chromosome are possible (or worth it) to distinguish and treat differently, and the form of the fitness function used to select "good" chromosomes from each generation. The meta-architecture structure for the SoS problem here was already selected; with one small exception (for the communication initialization, discussed later) there are no privileged gene components in the SoS meta-architecture. The remaining driver to a solution is the choice of fitness function. The fuzzy logic system approach is well suited to the type of judgments made about "good" SoS architectures [82], but certainly not the only approach, as discussed in the next section.

Most of the recent literature on fuzzy systems deals with treating uncertainty explicitly with Type II fuzzy systems. Type II systems treat the shape of the membership function edges as an additional parameter in fitting a solution. There is a contention that adding parameters to Type I fuzzy systems is equivalent to the extra degrees of freedom Type II systems allow for describing solutions. Several of these concepts were used in the definition of the membership functions and variable maps from real world variables to fuzzy variables.

It is institutionally difficult to find discussions of what works and what does not work in SoSs. For one thing, they are relatively rare. In addition, DoD examples typically:

- do not follow the normal DoDI 5000 series management processes, so normal reporting is not always enforced, and therefore records are sparse
- are not programs of record, so there is less than normal oversight by watchdog agencies
- frequently begin in an ad hoc way or as quick reaction efforts, so if they don't work, they are simply abandoned quickly for something more promising
- are occasionally classified or have significant classified components.

All these facts make it difficult to perform case studies on SoS lessons learned

Commercial efforts frequently fall under proprietary disclosure rules. Another reason we do not have good post-mortems on problem projects is that those stories can be embarrassing to those most likely to know what occurred. Finally, it is often simply the case that no one knows why projects fail or succeed. It may be simply that it was an idea whose time had come when it works.

2.5.4 Multi-objective Optimization

The architecture optimization in general is a constrained (e.g., by design requirements and restrictions), multi-objective optimization on a discrete design space. Optimization models used in the architecture search enable "moving from one configuration to the other in an ongoing search for better solutions,

but more importantly it is established with the aim of control and guidance" [83]. Multi-objective optimizations involve two processes, a search process that can explore all possible design parameters and a selection process that chooses good designs that constitute a compromise of several different objectives.

Depending on when the preference for each objective is expressed, multi-objective optimization methods can be broadly classified into two categories: decision making before search methods (also known as *a priori* approach or scalarization approaches), and search before decision making methods (also known as *a posteriori* or Pareto approaches) [84] [85]. As summarized in [83] [86] [87], examples of scalarization approaches include weighted sum approach, multi-attribute utility analysis, ϵ-constraint methods, compromise programming (non-linear-combinations), physical programming, goal programming, lexicographic approaches, acceptability functions, and fuzzy logic; examples of Pareto approaches include exploration and Pareto filtering, multi-objective genetic algorithms, adaptive weighted sum method, normal boundary intersection, and multi-objective simulated annealing.

The *a priori* approaches require designer to decide how to aggregate different objectives into a single objective function (also known as fix-up) before the actual search is performed [87]. Such approaches require *a priori* knowledge to make rational aggregation.

In *a posteriori* approaches, the search for optimal solutions is performed with multiple objectives being evaluated simultaneously, typically using the concept of "dominance" to rank solutions. Particularly, a solution x_1 *dominates* another solution x_2 if (1) x_1 is no worse than x_2 in all objectives and (2) x_1 is strictly better than x_2 in at least one objective [86]. The Pareto-optimal set is the entire set of non-dominated solutions among the search space, where the rest of the solutions are called dominated solutions [88]. Most Pareto-methods are concentrated on the approximation of the Pareto set [88] while keep the solutions diverse. "All elements in the Pareto-optimal set define reasonable solutions and are subject to further decision factors in order to choose a design for a given problem" [87]. In this manner an unbiased search can be performed. Moreover, Pareto methods also allow a single search to serve several problem-specific decisions without the need to repeat the search [87]. This feature gives Pareto methods an advantage over single objective methods because the designers are provided with a wide range of non-dominated solutions from which one or more solutions can be chosen. This post-search selection can be supported by further analyses using domain knowledge, additional problem information, or decision criteria, which are not necessarily formulated in the design task.

There are other classifications of optimization algorithms according to various considerations. For example, Cohon [89] classified them into the Generating methods and Preference-based methods (some known preference for each objective is used) based on whether Pareto-optimal solutions are generated or not. Hwang and Masud [90] and later Mittinen [91] fine-tuned Cohon's classification into the following four classes of methods:

(1) No preference methods are generating methods that do not assume any information about the relative importance of each objective. Instead, a heuristic is used to find a single optimal solution [86].
(2) *A priori* methods are preference-based methods that use information about the preferences of objectives *A priori* and usually find one preferred Pareto-optimal solution.
(3) *Posteriori* methods are generating methods where preference is used *a posteriori*. A set of Pareto-optimal solutions are produced by the algorithm. The decision maker then selects the most preferred one according to some further considerations.
(4) Interactive methods are preference-based methods that use the preference information progressively during the optimization process. It requires the interaction with the decision maker.

From the searching process perspective, optimization algorithms can be classified into either deterministic or stochastic (or heuristic) methods. Deterministic methods can be classified into gradient based methods (such as steepest descent, newton method, conjugate, penalty method) and derivative-free methods (such as simplex method, integer programming) [92]. Gradient-based algorithms can find local optima with high reliability and, in many cases, with high efficiency but might be trapped by local optima. Heuristic based algorithms can escape local optima and are stochastic in nature. They cannot guarantee the optimality of the solutions obtained and often yield different set of solutions each time they are run. No existing optimization technique is guaranteed to find the global optimum of a nonlinear, non-convex problem [93] [94].

No single optimization technique is applicable in general to all types of problems. The most effective way, however, to solve a given problem will always be dependent on the specifics and details of that unique problem [95]. A hybrid method that combines optimization methods in a complementary way may ideally both benefit from the relative strengths of each individual method and restrain its weaknesses.

In the case of architecture development, the design space could be exceptionally large thus precluding the use of brute force algorithms. On the other hand, deterministic algorithms that would be fast enough either might not exist or would be too complicated to define. Hence, the heuristic based search algorithms are more appropriate in such application, as they can find good enough solutions from a large design space within a reasonable amount of time with little or no reliance on the knowledge of the search space. Some heuristic based optimization algorithms that can possibly be applied to the architecture optimization are briefly discussed.

Hill Climbing (HC) [96]is an iterative algorithm that starts with an arbitrary initial candidate solution, then attempts to find a better solution by examining its neighbors (defined on the solution space). If a near neighbor with a better fitness value is found, a move to the new solution is made. Such “walk up the hill” process is repeated until no further improvement can be found. Such HCs are only guaranteed to find local optima. Near-global optima can be reached by using restarts (known as multiple-restart hill climbing), or more complex schemes based on iterations (e.g., iterated local search), on memory, (e.g.,

reactive search optimization and tabu search), on memory-less stochastic modifications (e.g., simulated annealing) [97].

Simulated Annealing (SA) [98, 99] is another local search algorithm exploiting neighborhood concepts. At each iteration of the search process, SA attempts to replace the current solution with a random solution chosen according to a candidate distribution, which is often sampled from the neighborhood of the current solution. The acceptation of the new solution is based on a probability that is a function of both the drop in fitness and a global parameter *T*. With this *T* parameter being gradually reduced during the search process, the SA can avoids local optima to a certain extent by giving more chances to less fit solutions in the earlier exploration stages but increasingly choosing the better solutions in the latter converging stages.

Tabu search [100] [101]is another meta-heuristic local search algorithm that proceeds by setting barriers or restrictions to guide the search process. Tabu search avoids being stuck at local optima by using tabu list which are a set of rules and banned solutions used to filter which solutions will be admitted to the neighborhood to be explored [100]. Then from the set of available moves, the best move is selected.. The application of tabu search in architecture related problems can be found at [102] [103].

Evolutionary algorithms (EAs), namely genetic algorithms, evolutionary programming and evolution strategies, have been recognized to be well suited to multi-objective optimization by virtue of the parallel search technique involved in addition to handling complex problems that involve discontinuities, multi-modality, disjoint feasible spaces or noisy function evaluations [104].

Among them, Genetic Algorithm (GA) [105] is one of the most used Evolutionary Algorithms (EAs).In GA, solutions (known as candidates, individuals or phenotypes) are encoded in a string form known as chromosomes (or genotypes of the genome). GA uses an iterative evolution process starting from a population of randomly generated candidates. In each generation, multiple candidates are stochastically selected from the current population based on their fitness. These candidates are then modified (by applying mutations, crossovers, or other reproduction operators) to form the offspring. The new population for the next iteration of the algorithm is produced from the offspring and the original population using a selection process. Many variants of this overall process exist, but the key ingredients i.e., recombination and selection guided by fitness functions, remain the same. EAs, as popular search techniques, have many applications in architecture related problems, for example, the architecture design [106] [107], formulation of predictive models of software projects [108] [109], and testing [110] [111]. Holland demonstrated the tremendous advantage of using genetic algorithms to certain optimization problems that can exploit its procedure [112]. However, MOEA sometimes perform poorly when there are many objectives to be optimized. To overcome this drawback, a new fitness evaluation mechanism is introduced in [113], which can continuously differentiate individuals into different degrees of optimality beyond the classification of the tratidional Pareto dominance. In addition, fuzzy logic is used to define a fuzzy Pareto domination relation.

On the other hand, various techniques have been proposed for handling of constraints in optimization problems involving evolutionary algorithms [114] [115]. These include penalty function methods [116] [117] [118], pair-wise comparisons [119], or simple comparison [120]. Other methods treat the constraint functions as one or more additional objective functions to be optimized [121] [122] Others combine multi-objective evolution with differential evolution [123].

The multi-objective evolutional algorithm (MOEA) is a popular Pareto-based optimization approach. MOEA is well suited for the architecture optimization. Research work in MOEAs in areas related to systems architecting include product design and product architecting [124] [125] [126] [127] [128] [129] [130] [131] . Most recently, genetic algorithms (GA) was also used as a methodology within a tool suite of agent base modeling (ABM) for system of systems (SoS) evolution, where the impact of constituent systems are thought not to be well understood with respect to the overall system of systems capabilities [35].

2.6 System Negotiation Decision Models with SoS

2.6.1 Selfish Negotiation Model

Resource constrained scheduling problems (RCSPs) concern with allocating scarce resources to activities over time in order to meet the requirements on the quality, quantity, completion time, and others in the delivery of projects, products, or services [132]. The problems are often formulated as optimization models where objective functions can be the minimization of project duration, minimization of the project cost given performance payments and penalties, and minimization of the consumption of critical resources [133]. Resource constraints can be on a period-to-period basis such as the amount of skilled labor available for each day; they can also be over the life of the project such as the total funding for the project. The solution to RCSPs specified when and how a job is processed. Various methods have been implemented to solve RCSPs, including integer programming [133], heuristics [134], genetic algorithm [135], and simulation [136].

RCSPs can be a candidate model of individual systems because they are relevant to the SoS problem in this research. Participation requests from the SoS can be seen as projects with multiple jobs. Each job has specified completion time (e.g., deadline), performance requirement (e.g., capacity requirement), and performance payments (e.g., funding). A project's jobs usually consume the same resources that are provided either by the project execution entity (i.e., the individual system) or the project provider (i.e., the SoS). For example, performing a project consumes a portion or all of the performance payment provided by the project provider; it may also need skilled workers belonging to the project execution entity. The project execution entity schedules all jobs over time by specifying when and how each job is performed. The best schedule is often determined using optimization techniques. That is, the project execution entity formulates an optimization problem for the project by specifying the objective of executing the project. Constraints must be considered include those specified by the project provider (e.g., project completion time), determined by the environment or context (e.g., the expected return from projects with similar risks in the market), and specified by the project execution entity (e.g., the minimum rate of return from the project). An assessment of the project based on the optimal project

schedule provides useful information for the negotiation between the project provider and the project execution entity. For example, if the project execution entity finds out that the performance payment provided by the SoS is not sufficient to cover project expenses or the project deadline is too short to be met, she will ask for additional payments or additional time with a justification.

2.6.2 Opportunistic Negotiation Model

Existence of individual systems within an SoS are abundant in many real-world scenarios. This includes selection from a number of transportation infrastructure projects [137], a bank/financial organization providing capital to different entrepreneurs [138], and a central artificially intelligent agent controlling a bank of robots [139], determining which robot(s) to use to assign a given task. The main idea in the opportunistic model is to capture the behavior of an adaptable agent, which is endowed with significant amounts of flexibility to perform a task and adjust to budgets [140].

Opportunistic model was based on an absorbing Markov chain [141] for modeling the duration of a project and gaging its budget. There is abundant literature on project management, which cites the need for mathematical models that can capture stochastic behavior of durations of activities and the resulting deadline extensions and budget overruns [142]. Code for the opportunistic model was written in MATLAB [143].

2.6.3 Cooperative Negotiation Model

Negotiations vary among agents in number of measurable and fuzzy issues. Quantitative issues can be in the form of measurable attributes such as funding, deadline and performance. Qualitative issues can include the behavior of the negotiation party and market induced effects [64]. Each successful negotiation necessitates resolving a range of such issues to the agreement of both parties. Agents usually are able to make adjustments between issues by prioritizing one over another (e.g., faster completion time with lower quality in order to come to an agreement). On the other hand, many system agents have the same capabilities but they vary along the dimensions of performance quality, cost of acquisition and availability within a specified time. Negotiations between automated agents typically have four prominent characteristics [144]. These characteristics are mentioned below along with the specific details that are pertinent to this cooperative model.

According to [74]cooperative negotiation in multi-agents is a decision process for resolving multiple issues, which may or may not be mutually exclusive. Game theory postulates cooperative negotiation as a non-zero sum game along multi-dimensional issues. Cooperative multi-agent negotiation has found uses in maintaining real time load of a mobile cellular network [68], modeling complex physiological phenomena [69] and resolving air traffic conflicts efficiently [70].

(1) The negotiation protocol and strategies

The negotiation protocol describes the rules of encounter between the negotiation parties. A negotiation protocol can handle a single issue or multiple issues [145]. The negotiation strategy is a specification of the sequence of actions (usually offers or responses) that the agent plans to

make during negotiation [146].The solution space of negotiation strategies is very large. Strategies are usually based on the nature of the behavior of the agent and its opponent or teammate. Negotiation strategy tries to model the function (or a set of rules) for proposing the values of multiple issues at each point in time [73]. The strategy used for a particular agent might turn out to be a poor choice for another. A static approach can also decrease awards after a number of negotiations. Therefore, an agent can learn by adapting based on rewards, as opposed to trying to model the other agents. Other considerations include trust of the agent on other agents and commitment involved in pursuing a negotiation [74].

(2) The information state of agents

Agents are classified based on information possessed at the time of negotiation into complete or partial information states. If the agent has the complete information of the environment, which includes the opponent agent's, negotiation strategy, the external factors that affect the negotiation and the effect of the agent's strategy on the opponent it is said that that agent is in a complete information state. Otherwise, if any information is unclear or missing the agent is assumed to be in a partial information state. Information in multi agent systems are comprised of utility functions that the opponent agents use to evaluate various attributes, the reasoning models of opponent agents, and the constraints of opponent agents.

(3) The negotiation equilibrium

In a symmetric situation, the negotiation solution does not depend on which agent is called agent one. This model is not symmetric. Both negotiating agents have different outlooks and different functions to estimate their utilities.

Furthermore, to estimate a utility for an issue an agent needs to define certain metric for calculating the values based on the information received. The metrics follow the criteria of John Christian [29], which has seven characteristics:

1) Relates to performance
2) Simple to state
3) Complete
4) States any time dependency
5) States any environmental conditions
6) Can be measured quantitatively
7) Easy to measure

The utility functions can be classified into linear and nonlinear. Agents that utilize linear utility functions can aggregate the utilities of the issue-values by weighted linear summation. However, such an approach is considered naïve for modeling real world scenarios as aggregations are unrealistic [76].

Multi-attribute utility theory (MAUT) [77] believes that each outcome issue or attribute is independent. MAUT proposes to have a separate utility function for each of the attributes. These utility functions can be aggregated together or the agent can negotiate each issue separately based on the values.

3 Multi-Agent based Architecting Model for Acknowledged SoS

The proposed Acknowledged SoS architecture is a construct comprising of several complex adaptive systems (CAS) which are evolving and adapting in space and time constantly [149]. ABM is an attempt to simulate and analyze complex adaptive systems. The agent based model (ABM) structure is developed to provide an Acknowledged SoS manager a decision making tool in negotiation of SoS architectures during wave model cycles. The ABM provides the capability to include the above mentioned multiple SoS meta-architectures generation modles namely Fuzzy –genetic optimization model, Multi-Level Optimization model, and Multi-objective optimization model. ABM also provides alternatives for participating systems to chooses rom three negotiation Models. The negotiation model for SoS is fixed. This approach helps create multiple alternatives to generate architectures for acknowledged SoS.

3.1 Agent Based Model Framework

Agents are governed by a collection of stimulus responsible rules. Rules are a method to describe the agent strategies in the environment they inhabit. The objective of ABM is to determine how both the communications and diverse activities of individual agents can create organization and pattern. The aptness of using an ABM model for this application area can be summarized in the following properties that define it:

i. ABM is able to demonstrate the traits of the complex system in a changing phenomenon;
ii. Agents are interdependent;
iii. ABM is supple enough to adjust to varying conditions

ABM attempts to bridge the gap, by transforming incoming information into meaningful and precise predictions of future states, in a dynamic command and control environment. ABM techniques will be utilized to simulate artificial negotiation intelligence. This intelligence emerges from the association and competition of many individuals and appears as harmonious decision making on the macro-level.

As illustrated in Figure 4, The overall ABM consists of 3 major elements; SoS acquisition environment, SoS agent, and a system agent. Each agent has its own set of behavior patterns. The ABM has one instance of the SoS agent and multiple instances of the system agent. The number of instances of the system agent corresponds to the number of systems in the SoS. The SoS manager instance and the individual system instances are embedded in the SoS acquisition environment model and are influenced by the changes in this environment.

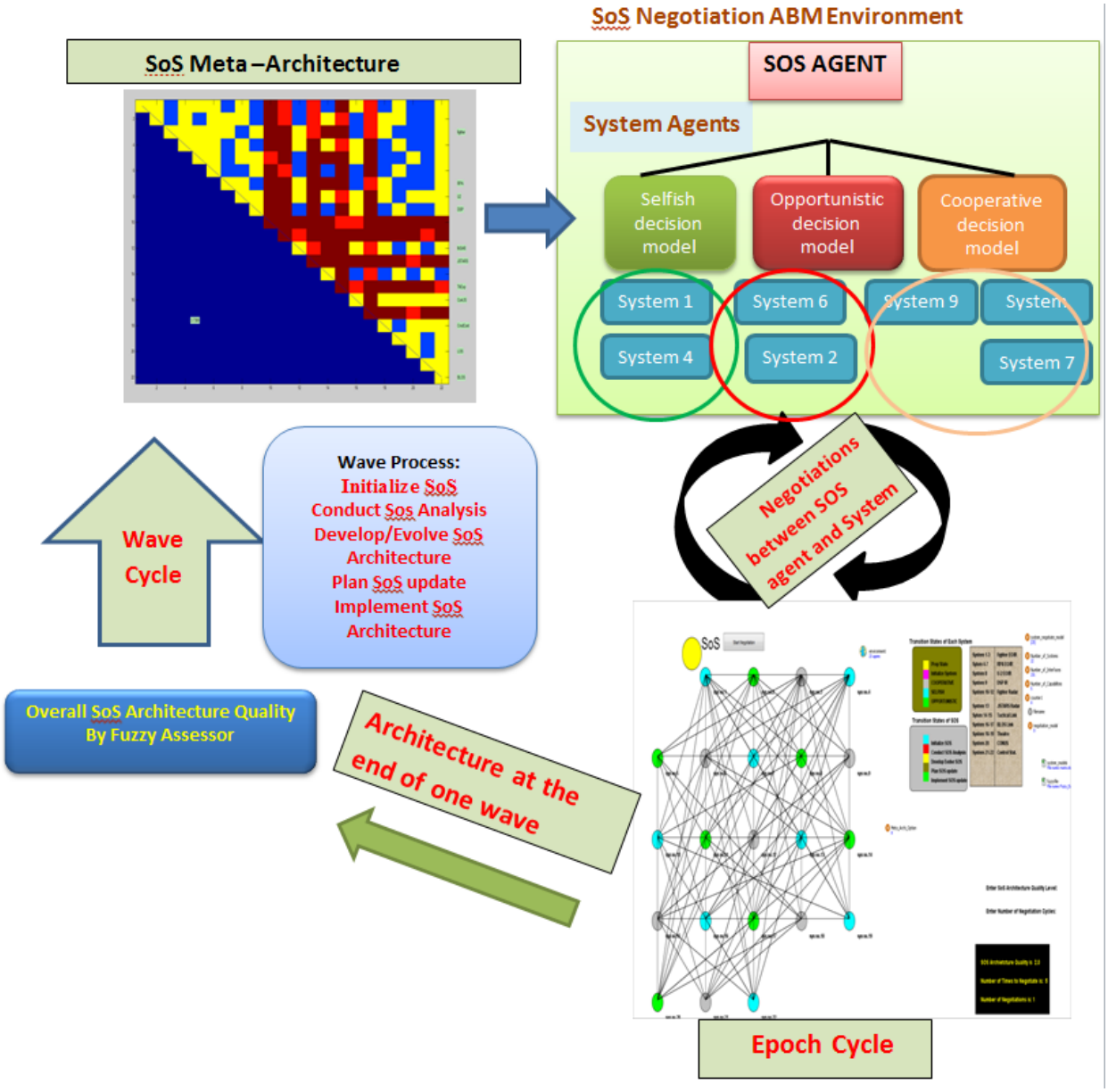

Figure 4. ABM framework for Acknowledged SoS

This model proposes a hierarchical architectural framework to support Acknowledged SoS architecting and analysis. The goal is not simply to reach a settlement, but to tackle frequent systemic problems and adapt to the changing conditions and relationships among all participating systems. The individual systems have limited or no information about other participating systems. All System agents are autonomous and assume that other agents also take decisions rationally based on their own internal state. The first level of the architecture the ABM provides an ability to include multiple SoS meta-

architectures. The meta-architectures include many sets of combinations defining systems' participation and their interfaces. To generate the meta-architectures, the SoS Manager organizes domain specific information. Information from the meta-architecture is analyzed and tabulated to come up with the required guidelines for individual systems. The SoS agent is responsible for generating SoS meta-architecture based on these input parameters and evaluating the overall quality of the SoS architecture. The SoS agent then negotiates individually with each of the collaborating systems based on the current guidelines. The second level of the ABM architecture involves the negotiation models of System agents. Each individual system has the capacity to execute one of three multiple criteria optimization models. System agents act on the basis of bounded rationality. This means that systems attempt to be rational while simplifying their decision making.

The SoS agent behavior is abstracted based on the Wave Process model. Each instant of the individual system agent has its own decision model which can exhibit selfish, opportunistic or cooperative behavior. The ABM simulates the negotiation dynamics among SoS agent and individual system agent in order to analyze the SoS architecture evolution summarizes the overall model architecture. Each round of negotiation between the SoS Aquistion Manager and System agent is called an epoch. Once the SoS Manager successfully negotiates with all system agents, it is able to define the overall SoS architecture based on the SoS negotiation model. The SoS negotiation model is further explained in the report. This process of arriving at a final SoS architecture completes a wave of the model. SoS agent can predefine the maximum number of negotiations it can have with each system agent. Similarly SoS agent can fix the minimum value of SoS architecture quality that it wants to achieve. If the threshold of number of negotiations is exceeded or the overall SoS architecture quality is not up to the already decided value the SoS agent regenerates a meta-architecture. This starts a fresh round of negotiations with the individual participating systems.

The implemented framework of the ABM is described later in the report in the main elements of the model such as Implementation of Multi- Agent Based Architecture Model for Acknowledged SoS

3.2 Fuzzy Genetic Optimization Model

Genetic Algorithm allows the exploration of large discrete design space and can avoid trapping by local optima. Therefore, it is well suited for exploring architecture alternatives. Fuzzy logic based approach is good at linguistic computation that is capable of quantitatively evaluating the set of possible solutions based on the decision-maker's preferences even in the presence of incomplete, subjective and ambiguous information. By combing these two techniques, an automated architecture generation, evaluation and selection process can be achieved with the goal of finding new-optimum candidate architecture. This section presents the key aspects of the fuzzy genetic optimization approach, which include (1) the representation of SoS meta-architecture as chromosome and the generation of architecture alternative using genetic operators (i.e., mutation and crossover) (3) the methodologies for evaluating SoS domain model.

3.2.1 Representation of SoS meta-Architecture

The meta-architecture for the SoS can be represented as a binary string **X**, consisting of a one if a system is participating, and a zero if the system is not participating. The first ***m*** bits represent the systems potentially in the SoS. The next ***m-1*** bits are the interfaces of the other systems with system 1. The next ***m-2*** bits represent the interfaces with system 2 (interface 1-2 is already represented), and so on down to the last bit representing the interface of system ***m-1*** with system ***m***. The meta-architecture represents the ***m*** systems' participation and the interfaces between them as binary string of length

$$\boldsymbol{m}\ \text{systems} + \frac{\boldsymbol{m(m-1)}}{\boldsymbol{2}}\ \text{interfaces} \quad = \quad \frac{\boldsymbol{2m + m(m-1)}}{\boldsymbol{2}} \quad = \quad \frac{\boldsymbol{m(m+1)}}{\boldsymbol{2}}$$

The meta-architecture allows any combination of ones and zeroes in a string of length ***m(m+1)/2***. An architecture instance is one chromosome of binary bits representing the systems and interfaces that are participating in the SoS. An instance is one particular string, represented as **X**. The presence of the individual system or interface is indicated by one bit of the string $\mathbf{X_i}$. When using the genetic algorithm (GA) approach, the string is called a chromosome. The subscript on the chromosome may also indicate one string in a population of chromosomes; it is clear from the context whether an individual bit of a chromosome or a single chromosome from a population is intended.

Since the linear representation of the chromosome for more than about ***m*** = 5 systems is difficult to display (or to locate individual interface bits), the upper triangular form is introduced. The systems are along the diagonal, and the interfaces are in the row and column intersections. This has the advantage that $\mathbf{X_{ij}}$ in the standard matrix element numbering is the interface between system i and system j. For the systems (not the interfaces) as represented in the triangular form, we occasionally use the notation $\mathbf{X_i}$ when $\mathbf{X_{i=j}}$.

For chromosome representation presented above, the architecture alternatives can be easily generated using traditional genetic operators, such as mutation, crossover and transposition.

Table 1 Chromosome in linear array form. The meta-architecture allows any combinations of ones and zeroes

X_1	X_2	X_i	...	X_m	$X_{1 \text{ with } 2}$	$X_{1 \text{ with } 3}$	$X_{1 \text{ with } m}$	$X_{2 \text{ with } 3}$	...	$X_{i \text{ with } j}$	...	$X_{(m-1) \text{ with } m}$
Systems					Interfaces							

X_1	X_{12}	X_{13}	…	X_{1i}	X_{1j}	…	X_{1m}
	X_2	X_{23}	…	X_{2i}	X_{2j}	…	X_{2m}
		X_3	…	X_{3i}	X_{3j}	…	…
			…	…	…	…	…
				X_i	X_{ij}	…	X_{im}
					X_j	…	X_{jm}
						…	$X_{(m-1)m}$
							X_m

Figure 5. Upper Triangular form of the chromosome, systems along the diagonal

3.2.2 Domain Model Representation

The SoS agent delivers individual plans to each system with proposed participation and interfaces in each capability, and proposed budget, performance contribution, and deadline for delivering the capabilities. The four attributes are evaluated at the SoS level, but negotiations about individual systems' participation are at the systems' level. Not until the SoS agent has the answers from all the systems is the new chromosome worth evaluating again against the same initial attribute list. The domain model is represented as the performance, budget and deadline in each capability, from each system, and for each of its interfaces. We are currently using a simple extension of a system capability to each interface, but there is room for adjustment to this quick approximation. The system budget is simply the sum of estimated development costs for each change or interface added by the SoS manager suggested architecture found through the GA approach using the initial fuzzy assessment with estimated values. During the ABM negotiation phase the systems propose new, better grounded estimates of the performance they can provide, funding they will require, and deadlines they can meet. Although the SoS manager has estimates for these things by interface, the expected performance, offer of funding, and deadline is rolled up to a total per system. The system agents may divide up their totals as they think best, with theoretically better knowledge of their ability to deliver capability to the SoS.

The capabilities **C** of the SoS arise primarily from the summation of the capabilities inherent in the systems. We model ***n*** capabilities possible in each system. Most systems bring only a few capabilities. When a system interfaces with another system with different capabilities, they gain some portion of those non-organic capabilities through the interface. This is the key value of the SoS. The capability vs. system matrix is also binary, indicating only existence or not existence of each capability within each system.

$$C_i \quad i = 1 \ldots n \;\; Represent\ the\ various\ capabilities$$

$$C_{ij} \quad i = 1 \ldots n, \quad j = 1 \ldots m \quad As\ the\ capabilities\ of\ each\ system$$

is the matrix of up to n capabilities contributed by each of the m system

A system *type* identifies with the primary capability it has. The performance of each type of system is represented as a relative number compared to the other types of systems. Performance could be assigned for each capability, but to simplify the model, only the major capability of each system is accounted for. The performance is given in a matrix as follows:

$$Perf\, i \quad i = 1 \ldots m \quad Performance\ of\ each\ system\ in\ their\ major\ capability$$

There is a cost for developing, installing, testing and training to use the interface, and another cost for the operation of the system within the SoS. The estimated costs for interfacing between systems and operating the systems in the SoS are given in terms of the proposed funding in the following two matrices:

$F_{dev\ j}\ j = 1 \ldots m \;\; Funds\ for\ each\ system's\ development\ of\ \ an\ interface$, and

$$F_{ops\ j}\ j = 1 \ldots m \quad Funds\ of\ \ operating\ the\ system\ within\ the\ SoS$$

For the development of either a system capability that does not exist already, or the development of an interface to make the SoS possible, there is a schedule; the SoS manager proposes a deadline for the system to develop the each interface. Capabilities are not normally developed in an acknowledged SoS because such development typically require many years. The interfaces are normally easier, quicker, and less expensive to develop, while the systems and their major capabilities are "come as you are." The estimated deadline for development is given in a matrix

D_i i = 1…m *The deadline in units of epochs for developing an interface*

3.2.3 Methodology for Evaluating SoS Domain Independent Information

In DoD acquisition management, a program's (system's) current status is often presented as a color code, as in the Contractor Performance Assessment Report [150]. These reporting methods primarily allow reviewers to compare evaluation of alternative architectures (early), or a program status (later in the development cycle), from one quarterly review to the next, or between programs. The single color summary glosses over many fine details to get to the attribute summary color or single word description that summarizes the state of the entire program. An overall evaluation, that combines several attributes, is still largely a gestalt of the component attribute values, especially if different areas are

weighted differently. Attribute evaluations themselves are well suited to fuzzy logic approaches because of the difficult nature of boundaries between subjective evaluation ranges, and the ability to encode simple rules to combine values from different areas. These match well with fuzzy, linguistic approaches to evaluation. A particular SoS architecture (or program status) may fall partially into an Acceptable, and partly into a Marginal set by summing many underlying status components, just as any point on the quality scale between 2.3 and 2.8 does. This region of confusion, ambiguity, or uncertainty can be due to

- Differing interpretation of components' status facts by the reviewing subject matter experts (SMEs) based on their experience and understanding of the presentation
- Unrecognized (and typically bad) trends within the numerous components of any summary evaluation
- The energy of the presenter of the facts may influence the impression of those facts in the reviewers' minds
- Unconscious or even conscious bias of the evaluators.

The normal method of accounting for this in program management is to have multiple reviewers and do some kind of averaging and/or throwing out high and low values with room for discussion among the reviewers, and having a head judge (program manager) responsible for managing to a consensus and over ruling dissent if necessary. This is what the fuzzy associative memory or fuzzy inference system does by summing the contributions of partial membership functions at an evaluation point in the judgment range. One of the advantages of the fuzzy approach is that the rule base for combining values in different areas can be made small, easy to understand, and simple to evaluate. This approach can get out of hand, because the size of the rule base increases exponentially when many areas are evaluated and combined. [81] Because we were trying to elicit general principles, we kept the rule base very small. A smaller rule base with few exceptions improves one's ability to explain the SoS quality to stakeholders as well as to one's self.

The equivalent of color codes from red, yellow, green, and blue, were proposed for the granularity of the assessment, but to make it more general, called them: "Unnacceptable," "Marginal," "Acceptable," and "Exceeds." One could easily use different scales, with different numbers of discreet granulations if it proved useful or normal practice in a domain. The principle is demonstrated here with four granulations in each dimension, for both input and output.

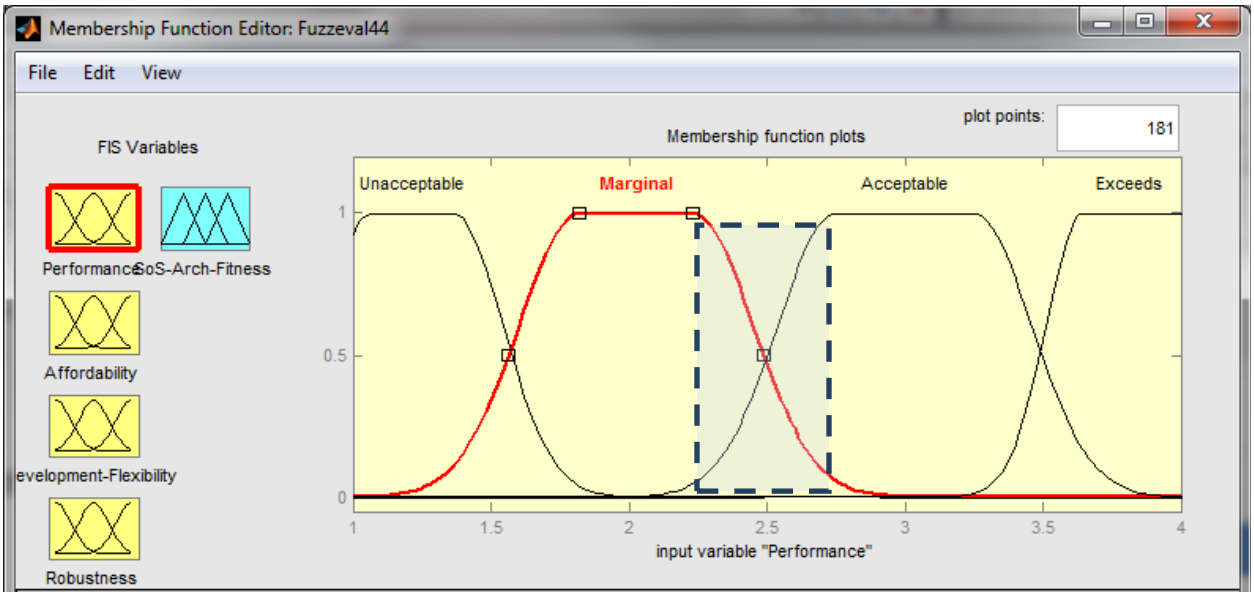

Figure 6. Representative Fuzzy Attribute Value Membership Functions showing overlap at edges of granulations

The attributes can be domain dependent. How you measure each attribute, and the criteria for valuing each one, for example, between acceptable and marginal are domain dependent. The universe of discourse is the total range of all the attributes. We mapped the domain independent range from 1 to 4, that is, from 1 = Unnaccepable to 4 = Exceeds, with each of the individual values being "around" the integers between 1 and 4. The shape of the membership functions account for the uncertainty of being on the cusp between two values. The color codes, or the terms (e.g., 'Marginal') are linguistic variables, describing the degree of membership in the fuzzy set that contains all programs or SoS architectures, or attribute values in that same range. There is a domain dependent mapping of measurement values to the domain independent fuzzy membership functions.

The steps to create a SoS domain model for experimentation are conceptual, largely independent of the domain. However, when working a real problem, the values take on meaning within the domain.

- The meaning and ranges of attributes against which the SoS is to be evaluated must be identified, defined, the range of the granulation and regions of overlap between membership functions decided, and the attributes checked for the degree of independence or correlation between them. Having correlation between two attributes decreases the value of the similar attribute to discriminate among potential solution architectures.
- Methods to evaluate the SoS against these attributes must be developed, assessed, and most importantly, agreed to by the stakeholders. For the study of SoS, we must assume some kind of advantage from joining different systems together. Otherwise, it does not make sense to organize a SoS.

To accomplish the above steps, a series of guided interviews with stakeholders are conducted by subject matter experts with architecting skills. Some of the required information is gleaned by simply letting stakeholders talk about their needs, their imagined or desired solutions and the characteristics of those solutions. Some information comes from guided questions exploring the stakeholders' needs and current operations, state of technology, capabilities of existing systems, impact of changes to existing hardware, software, procedures, processes, tactics and training, and outside influences such as market growth, competition or threats. This is similar to conducting the standard strengths, weaknesses, opportunities and threats (SWOT) analysis from business school. There are many methods for constructing the membership functions in the literature, but they all require domain knowledge to formulate the problem, construct the questions or guide the discussion among the stakeholders. [82] [151]

- Several trial methods of mapping issues, features and attributes to architecture structures may need to be created, tested and discarded before finding a good one
- Validity and range of applicability of existing and new models for predicting performance, cost or other attribute values are explored
- Relevance and usefulness of existing performance models are compared with proposed simplified models
- The process must result in a well-defined model for use in the SoS analysis that discriminates between proposed architectures selected from the meta-architecture

The purpose here is to have a reasonably simple evaluation process that depends on binary decisions about participation and interfaces between systems (and capabilities) in the SoS, as represented in the ones and zeroes of an individual chromosome. It must also correlate with some impact to the SoS quality (fitness) through changes in the participation model (the chromosome). Then it can be used in the fitness function for the optimization (or at least selection of a near optimum architecture) process of the GA. Success in the GA means a chromosome can be handed off to the ABM to exercise the system behavior and cooperation rule models. Our fuzzy modeling subgroup of SMEs acted as both interviewers and stakeholders for the development of the domain *dependent* models we used for the two SoSs, discussed in section 4.2.

To summarize, our framework includes four fitness evaluation categories, from unacceptable (1) to exceeds (4), for four SoS attributes that are almost generic, but quite tunable for many selected domains. These are **Performance, Affordability,** developmental **Flexibility,** and **Robustness** in operation.

- The Systems bring Capabilities that are summed to provide SoS **Performance**
- There are costs to develop interoperability, and costs to operate the systems. These both are summed to create the inverse of **Affordability**
- The SoS manager would like to have options (**Flexibility**) in developing each SoS capability, so she dislikes single system sources for a required capability

- Since the SoS might not have the desired and prepared systems at the time of need, **Robustness** is measured as the relative loss of performance when a desired system is unavailable.

Generically, this set of attributes comes close to satisfying the method characteristics noted above in section 3.2.3. They are not highly correlated. More systems provide more performance, but also more cost, so affordability goes down but not in a deterministic way because different systems have different costs and different performances. Adding more systems that are inexpensive may not drive cost up as much, while additional expensive systems might not increase the performance as much as expected if everything were expected to be linear. Having multiple sources of each capability is slightly related to the total number of systems. However, not highly correlated because as soon as one has more than two systems for one capability, the improvement stops. Robustness with this definition is slightly correlated to flexibility, since having multiple systems for each capability does correlate with having multiple sources, flexibility only counts the number of systems, while robustness counts the performance gain or loss from losing one system. The effect of the NetCentric bump (see below) changing performance in a noisy way depending on the feasibility of the interfaces largely decouples that possible correlation.

Other SoS attributes can easily be conceived: Degree of risk in herent in the architecture, amount of manpower required by the new concept, amount of interference with the remaining missions of the systems, etc. We selected the four attributes we did because there were more obvious connections to the participation model of the meta-architecture.

All the discussions so far would yield a model that is fairly linear. To make the model both more realistic and more interesting to analyze, an interoperability 'bump' is introduced. Recalling a tenet of NetCentric Warfare [152], a favorable performance impact from improved information sharing between the systems is postulated. This is proposed as a modification to the simple sum of performance from each system. The sum is multiplied by a small 'bump' factor for each *valid* interface between the systems. The factor is proportional to the count of feasible interfaces in the meta-architecture framework. The affordability is also multiplied by this factor. The major capabilities are delivered by the systems, but the SoS performance and affordability is multiplied by a factor depending only on the existence and use of the interfaces.

For the independent systems that make up the class of SoS under consideration, the interfaces they might have occur through a 'communication' system (or a transshipment point if the resource being exchanged through the interface is not information). When communication systems are included as one of the *types* of systems in the chromosome, it is a closer model of a real world SoS. The way the communication system is used as part of the interface is as follows:

If $\mathbf{X_{ij} = 1}$ for **i** and **j** which are **not** communications systems, then we mark that interface as a ***feasible*** interface **only** if $\mathbf{X_{ik} = 1}$ **and** $\mathbf{X_{jk} = 1}$ **and** $\mathbf{X_i = 1}$ **and** $\mathbf{X_j = 1}$where ***k*** is the index of a communication system.

If a system is not participating, then trying to interface with that system is **infeasible**. This does not prevent another system from spending funds, effort and time to develop that interface; this is an

attempted interface, but you need both systems to participate. In the same way, if two systems cannot communicate over a common communications system, then the interface between them is also ***infeasible***. This concept is shown below in where the chromosome might show a '1' for an attempted interface, but either the other system or the communications paths make the interface infeasible. There is no distinction if the infeasibility arises during development or operation in the simplified model.

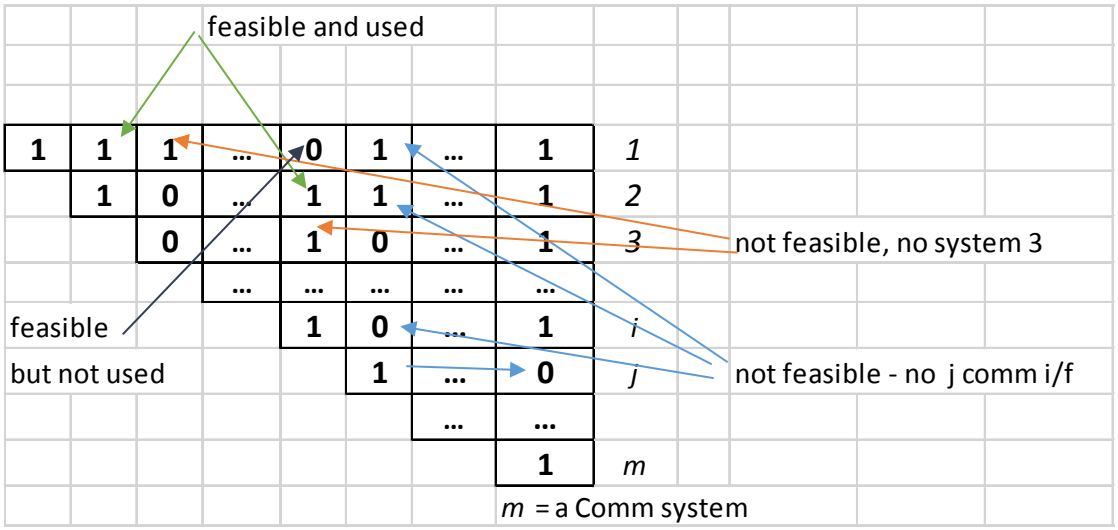

Figure 7. Illustrating the concept of feasibility of interfaces through a communication system

Regardless of the type of SoS, the fact that independent systems are used together almost requires that the interfaces occur through a communication type of system. That is, if system A wants to interface with system B (and neither are communication systems), then they must both interface to a common communications system. Examples of this might be a type of radio or digital communication channel, or some sort of transshipment point for a transportation SoS. For example, an air freighter cannot transfer cargo directly to a ship or vice versa. They must transfer cargo to some other type of system that moves between them, because they cannot be physically in contact between the taxiway and harbor. However, a properly equipped seaport might be able to transfer containers directly from shipboard to a train via a crane that can reach both. In our example domains introduced in section 4, two aircraft, or an aircraft and an exploitation center can transfer data via a radio communications link. The systems both would have to be modified to carry, and to use, the common communications link. It is also useful to penalize the performance for using *infeasible* interfaces. In the GA approach to a solution, the algorithm must be able to return a fitness value for any random chromosome selected from the meta-architecture. The penalty helps select chromosomes with useful interfaces. In a real case, one could make arguments for an appropriate value to be used for the bump, the reward, and the penalty. Experiments with the GA, using the fuzzy fitness function, were able to determine sets of factors that

allowed selection of relatively 'good' chromosomes to be handed over to the ABM through the GA for the negotiation process..

The "feasibility" test for interfaces is described as follows: if two systems think they have an interface by virtue of having a one in the ijth interface position, it is an infeasible interface unless they also have a common interface through at least one of the communications system links. Heuristically, the bump is set to approximately double or triple the performance or affordability, based on the largest possible number of interfaces, if they were all feasible. The feasibility test itself is only slightly correlated with the other attributes. The size of the positive and negative bump is a parameter that can be adjusted for the domain and number of interfaces possible in the SoS. Heuristically, the bumps are small, about a few tenths of a percent for each interface. They couple to the performance and affordability through the interfaces as follows:

$$Perf_{SoS} = (Perf_{\sum systems})(1 + bump)^{(\sum feasible - \sum infeasible)} Perf_{final}$$
$$= Perf(1 + bump)^{\sum feasible - \sum infeasible}$$

$$Afford_{SoS} = (Afford_{\sum systems})(1 - bump)^{(\sum feasible - \sum infeasible)} Afford_{final}$$
$$= Afford(1 - bump)^{\sum feasible - \sum infeasible}$$

The bump value provides an opportunity to do sensitivity analysis on the value of interoperability. One could also have different size effects on performance and affordability, or any of the attributes. In this project, on the intention was to demonstrate that the concepts could be made to work together, not to explore the full range of possible applications.

Overlaying the feasibility factor arising from the requirement to use the communication systems to achieve an interface required the use of color. The upper triangular form of displaying an example ISR chromosome with a colored block for each "one" of the string, and a brown block for each "zero" that would have been infeasible with our definition, is shown in The red blocks are "ones" that are not feasible, the yellow blocks are "ones" that are feasible, and the light blue blocks are "zeros" that are feasible – in other words, a feasible interface through a communication system is possible, but the chromosome shows it is not used. The total used/feasible systems and interfaces is 29, the total used/infeasible (invoking a performance and affordability penalty) is 94, the total number of unused but feasible interfaces is 17. The dark blue background color is unused – not in the upper triangular matrix.

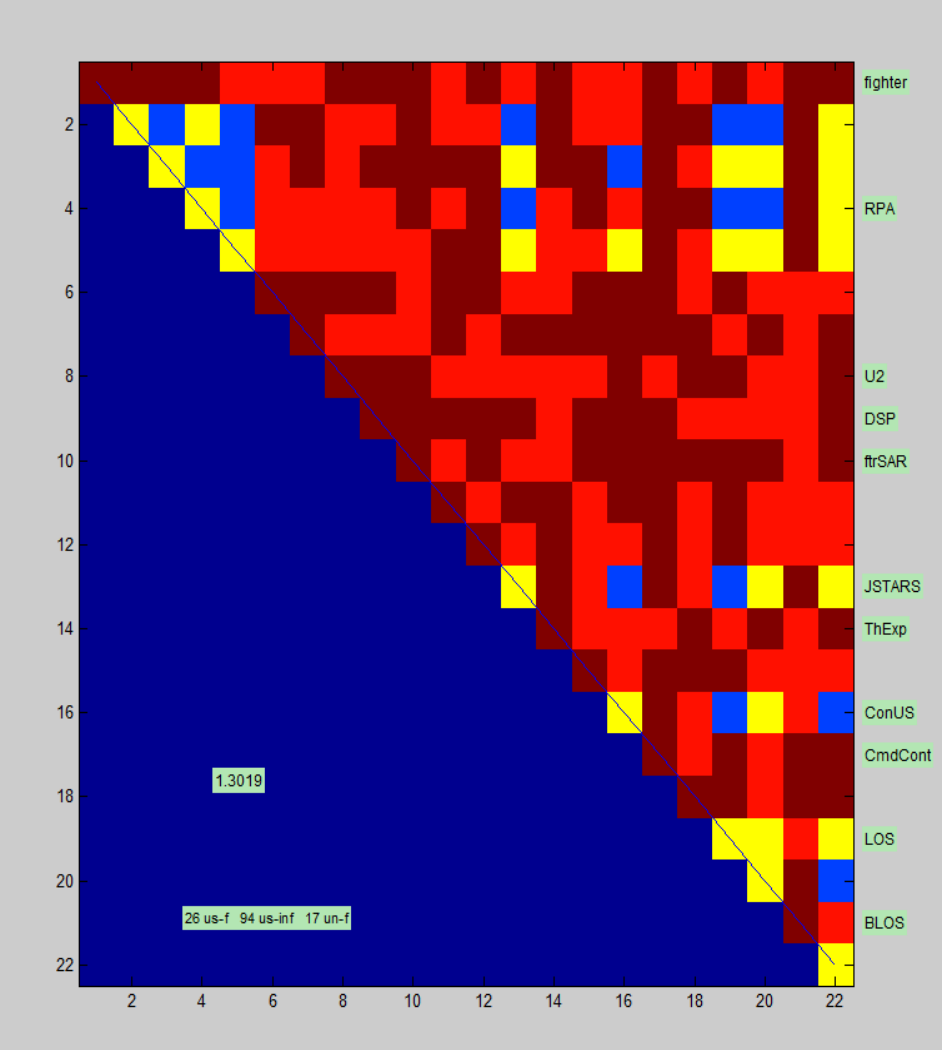

Figure 8. A chromosome (upper triangular form) with feasibility displayed and system types labeled

The capabilities of the SoS are primarily brought by the systems. The interfaces do not contribute any specific capabilities by themselves, but enhance the systems' capabilities. To make the GA work properly, one should not have extra, unique checks of certain parameters within the chromosome depending on the domain, but do all selection using only the total 'fitness' of the chromosome. Therefore, using the feasibility concept, there is a *penalty* added for using an *infeasible* interface. The sizes of the reward and the penalty are adjustable. Example values were found to show it could work as advertised. In a real problem, one would expect to discover these values inherent in the problem definition. The penalty and reward factors multiply the sums that make up the Performance and the Affordability.

The Flexibility attribute describes the number of choices available to the SoS manager. If a proposed SoS chromosome is limited to one system as the only source of a capability, then the SoS manager is

severely limited in flexibility during development. She must have that capability, so her ability to negotiate is reduced. On the other hand, if the each capability is available from more than one source, the SoS manager has flexibility during development of the SoS. For example, in the ISR domain example discussed later, we have an EO/IR search capability provided by several systems: Fighters, remote piloted aircraft (RPAs), U-2, or DSP can all provide EO/IR. Another Capability is side looking radar. This is also available from fighters (different ones than EO/IR equipped ones) and JSTARS. If a chromosome does not have at least two system sources of each capability, then Flexibility is reduced. The evaluation of Flexibility consists simply of counting the capabilities that have less than two sources. High flexibility for systems manager, hence for the meta-architecture is associated with having multiple systems able to provide at least some of every capability; high number of capabilities provided by only one system equates with low flexibility – in other words, the SoS manager has little flexibility in choosing among systems if she needs these capabilities and there is only one system to get them from.

The interesting thing about this Acknowledged SoS meta-architecture is that any system may decline to participate (or be operationally reassigned), and any interface can be there or not, depending on cooperation from the other system, the offer and negotiation with the SoS manager, and the environment. Even systems that fully cooperate in building the SoS capability may be called to higher priority task, or nullified by enemy action when it comes time to use the capability. Therefore, the SoS capability assessment on any particular day needs to be able to evaluate any combination of ones and zeroes. This point suggests one of the attributes: Robustness. We define the robustness as being greater if a smaller loss in performance results for the loss of any single cooperating system and its interfaces from the SoS architecture instance.

Summary of the Required Inputs to Create A Domain Specific Model

- The set of existing, potential systems
- The set of capabilities that could be contributed by each system through small changes, such as adding a new interface
- A short list of rules for combining the systems, interfaces, and their capabilities
- A short list of what feasible changes (typically adding an interface) could (or could not) be made
- The method by which each *contribution* from the systems and interfaces is *measured* against the desired capabilities (Key Performance Attributes – KPA)
- Estimated performance, funding and deadlines for each System in the SoS

Rules for Combining Attributes to an Overall SoS Fuzzy Assessment:

One of the modular approaches for suggesting an initial SoS architecture to the ABM negotiation is by combining the individual attribute evaluations is within a fuzzy inference system through a small number of rules we made for combining attribute values to an overall SoS assessment. Combining the various attribute measures to form the overall SoS assessment is relatively straightforward. For example, if half the attributes are slightly below average (one-step below Acceptable) and half the attributes are exceed requirements (one-step above Acceptable), then the SoS should average out to

Acceptable, with some better areas balancing out the poorer areas. This is assuming the attributes are equally weighted. Most of the rules are required only if there is an exception to the general rule just described. In the ISR example, better performance and affordability together outweighed poor flexibility and/or robustness. If all the attributes were above average, then that makes for an exceedingly good SoS, because this so rarely happens. Of course, if all the attributes are exceedingly good, then we let the general rule stand, and the SoS is exceedingly good. If one attribute is very good, but the rest are marginal, the good one can't pull the SoS up above marginal. Finally, if any of the attributes are unacceptable (the bottom rung of evaluation), then that drags the SoS evaluation down to unacceptable, too, regardless of how good the remainder of the attributes are. These rules are summarized in Table 2.

Table 2. Simple Fuzzy SoS Evaluation Rules

Plain Language Rule	Fuzzy Rule Definitions from MATLAB Fuzzy Toolbox
If ANY attribute is Unaccptable, then SoS is Unacceptable	• If (Performance is Unacceptable) or (Affordability is Unacceptable) or (Developmental_Flexibility is Unacceptable) or (Robustness is Unacceptable) then (SoS_Arch_Fitness is Unacceptable)
If ALL the attributes are Marginal, then the SoS is Unacceptable	• If (Performance is Marginal) and (Affordability is Marginal) and (Developmental_Flexibility is Marginal) and (Robustness is Marginal) then (SoS_Arch_Fitness is Unacceptable)
If ALL the attributes are Acceptable, then the SoS is Exceeds	• If (Performance is Acceptable) and (Affordability is Acceptable) and (Developmental_Flexibility is Acceptable) and (Robustness is Acceptable) then (SoS_Arch_Fitness is Exceeds)
If (Performance AND Affordability) are Exceeds, but (Dev. Flexibility and Robustness) are Marginal, then the SoS is Acceptable	• If (Performance is Exceeds) and (Affordability is Exceeds) and (Developmental_Flexibility is Marginal) and (Robustness is Marginal) then (SoS_Arch_Fitness is Acceptable)
If ALL attributes EXCEPT ONE are Marginal, then the SoS is still Marginal	• If (Performance is Marginal) and (Affordability is Marginal) and (Developmental_Flexibility is Marginal) and (Robustness is Acceptable) then (SoS_Arch_Fitness is Marginal)
	• If (Performance is Marginal) and (Affordability is Marginal) and (Developmental_Flexibility is Acceptable) and (Robustness is Marginal) then (SoS_Arch_Fitness is Marginal)
	• If (Performance is Marginal) and (Affordability is Acceptable) and (Developmental_Flexibility is Marginal) and (Robustness is Marginal) then (SoS_Arch_Fitness is Marginal)
	• If (Performance is Acceptable) and (Affordability is Marginal) and (Developmental_Flexibility is Marginal) and (Robustness is Marginal) then (SoS_Arch_Fitness is Marginal)

This simple set of rules demonstrated the fuzzy inference system's capability to treat the assessment non-linearly. The resulting 3D representation of the SoS fitness surface for two variables at a time is shown.

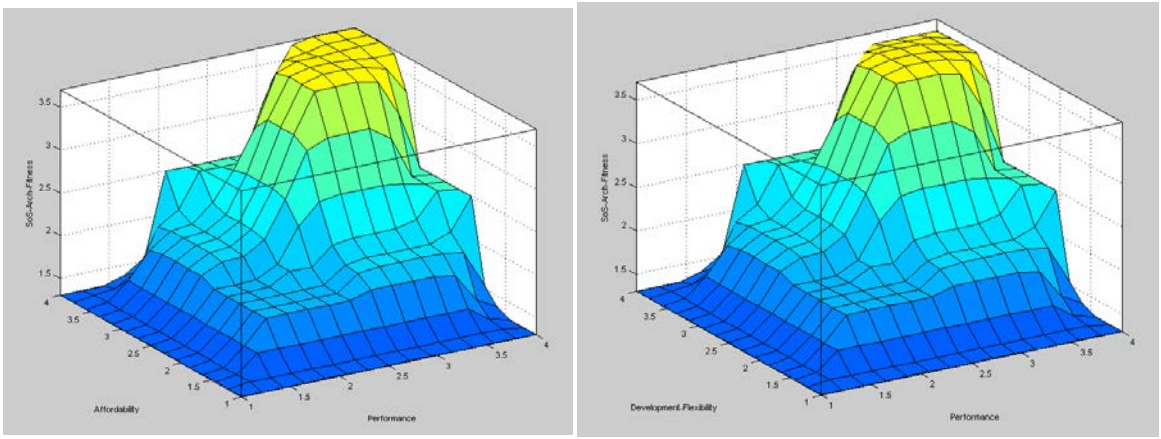

Figure 9. Non-linear fuzzy attribute to SoS evaluation mappings

3.3 SoS Negotiation Model

3.3.1 Game Theory Based Negotiation Model: Incentive Contracting

Game theory approach is applied to various economic problems including incentive-contracting problems in multi-agent settings [51]. In this setting, SoS manager is subcontracting its tasks to individual system agents. The problem for the SoS manager is to provide a contract and convince the individual systems (contractors) to join the SoS architecture and motivate them to do their tasks well. On the other side, for the individual system the success in carrying out the tasks depends on the time and work intensity with which it will put into fulfilling the task. This work intensity is referred to as effort level.

Since individual systems have their own motivations and objectives they want to do as little as possible and gain the highest rewards. The SoS manager's objective is different than the individual systems as the SoS manager wants to give lower rewards and obtain larger outcomes. This problem can be formulated as a multi-constraint problem where SoS manager tries to choose the reward for the individual system, so as to maximize its expected utility subject to two constraints [51]:

$$\text{Maximize } r_1, \ldots, r_n \sum_{i=1}^{n} \wp(\hat{e}, q_i) U^{SoS}(q_i, r_i)$$

With the constraints:

1. (Individual rationality)- Reward for the individual system must be large enough to motivate the individual system to prefer the contract rather than reject it. This means that expected utility of the individual system will be at least as much as its reservation price $\sum_{i=1}^{n} \wp(\hat{e}, q_i) U^{is}(\hat{e}, r_i) \geq \hat{u}$

2. (Participation constraint)- Provides the individual system with the motivation it needs to choose the effort level that the SoS manager prefers, given the contract it is offered.

$$\hat{e} = argmax_{e \in Effort} \sum_{i=1}^{n} \wp(e, q_i) U^{is}(e, r_i)$$

The following Table 3 summarizes the variable notations.

Table 3. Variable Notations used in the Game Theory Based Negotiation model

Notation	Meaning	Comments
Effort	Set of efforts of the individual system	$e_1, e_2 \ldots, e_i \in Effort$
Outcome	Set of possible monetary outcomes	$q_1, q_2 \ldots, q_i \in Outcome$ $q(e) \in Outcome$
Rewards	Set of possible monetary rewards to the individual system	$r_1, r_2 \ldots, r_i \in Rewards$ $r(q) \in Rewards$
U^{is}	Individual system's utility function	$U^{is}: Effort \times Rewards \rightarrow \mathbb{R}$
U^{SoS}	SoS Manager's utility function	$U^{SoS}: Outcome \times Rewards \rightarrow \mathbb{R}$
$\hat{u}$	Individual system's expected utility when it does not accept the contract	Reservation price
$e^* \in Effort$	Efficient effort level for the SoS manager	Given individual system constraints
$q^* \in Outcome$	Efficient outcome for the SoS manager	Given individual system constraints
$\wp$	Probability function (used for cases under uncertainty	$\wp: Effort \times Outcome \rightarrow \mathbb{R}$

SoS manager's utility function:

SoS manager utility function is in the form of Neumann-Morgenstern utility function [153] where preference over probabilistic outcomes is determined by maximizing the expected utility. This satisfies the rationality assumption in game theory based approaches. In the model, SoS manager's utility function, $U^{SoS}(q_i, r_i)$, is a function of the outcomes (performance) and rewards (funding).

Min-additive utility function is proposed in [154] which is a weighted combination of additive utility function and minimization over a set of single attribute utility functions. The min-additive utility function is suitable for SoS engineering as the SoS manager is trying the maximize the performance while minimizing the funding and deadline. The min-additive utility function is described briefly. Detailed information about the utility function can be found in [154].

Let $x = \{x_1, x_2, \ldots, x_n\}$ be the set of attributes that are interest to the SoS manager. Each attribute varies from the worst preferred value (x_i^{WORST}) to the most preferred value (x_i^{BEST}). Utility function expresses SoS manager's strength of preference for various levels of x.

Additive utility functions are suitable for outcomes that the SoS manager prefers, i.e. performance

$$ADD(u(x)) = \sum_{i=1}^{i=n} w_i u_i(x_i)$$

Where w_i non-negative weights that sum to 1 and $u_i(x_i)$ is a single attribute utility function such that $x_i^{WORST} = 0$ and $x_i^{BEST} = 1$

Minimization utility functions are suitable for unacceptable outcomes for the SoS manager, i.e. deadline and cost.

$$MIN(u(x)) = min\{u_1(x_1), u_2(x_2), \dots, u_n(x_n)\}$$

The min-additive utility function combines additive and minimization utility functions to benefit the advantages and offset the disadvantages of both functions.

$$MA(u(x)) = w_{MIN}(u(x)).MIN(u(x)) + w_{ADD}(u(x)).ADD(u(x))$$

$$w_{MIN}(u(x)) = 1 - MIN(u(x))$$

$$w_{ADD}(u(x)) = MIN(u(x))$$

$$MA(u(x)) = (1 - MIN(u(x)).MIN(u(x)) + MIN(u(x).ADD(u(x))$$

where $MA(u(x))$ represents the SoS manager's utility function, $U^{SoS}(q_i, r_i)$.

Levels of outcome, effort and rewards:

Different levels of effort result in different levels of outcome. The SoS manager evaluates outcome as a function of performance. Thus we can define different levels of outcome (performance) as $q_1, q_2 \dots, q_3 \in Outcome$ where$q_1 < q_2 < q_3$. For example, $q_1$is a marginal outcome for the SoS manager, q_2 is an acceptable outcome, and q_3 is an exceeding outcome for the SoS manager.

Since outcome is a function of the effort level of the individual system. We can define different levels of effort for the individual systems where $e_1, e_2 \dots, e_3 \in Effort$ where$e_1 < e_2 < e_3$. In a similar way $e_1$is marginal effort, $e_2$is acceptable effort, and e_3 is exceeding effort level.

Rewards are also a function of the outcome. Higher rewards are given for desired outcomes. Thus we can define different levels of reward for the individual systems such that $r_1, r_2 \dots, r_3 \in Rewards$ where $r_1 < r_2 < r_3$

In our case, effort level depends on the deadline and funding changes in the individual system. A change in the effort (Δd, Δf) will impact the outcome (performance) which will change the level of reward for the individual systems.

Determining different levels of performance and effort is domain dependent. Besides the SoS manager should take into account how change in performance, funding and deadline impacts the overall SoS architecture quality. In the ISR SoS domain, Fuzzy assessment is used to evaluate the overall SoS

architecture quality. Performance, affordability, robustness and flexibility are the four attributes to evaluate SoS architecture quality. This tool is used to determine values for various levels of performance and funding changes. The performance and affordability values in the Fuzzy assessment are changed incrementally to observe the deviation in the overall SoS performance and affordability. For example 3 levels of effort are determined using the Fuzzy assessor:

Exceeds performance and effort (q_3, e_3) : Δp=0, Δf=0

Acceptable performance and effort ($q_{2,}e_2$): Δp=20%decrease, Δf=60% increase

Marginal performance and effort: Δp (q_1, e_1) =50%decrease, Δf=100% increase

For deadline change, the number of interfaces for the individual system is calculated. The deadline should not exceed the total time allocated for all interfaces.

Individual system's utility function:

The SoS manager has limited information on individual system utility function. However, from a contracting perspective individual systems will try to maximize the reward (funding) with minimal effort level. A simple utility function that captures this relationship can be expressed as

$$U^c(r, e) = r - e$$

Individual system's reservation price:

The SoS manager has limited information on each individual system's reservation price which is the minimum reward that will make the individual system accept the contract. Otherwise, individual system can direct its resources to achieving its own objectives. The reservation price for individual systems is expressed as

$$\hat{u} = (f_{offer} - finitial) * w$$

where w is the weight allocated to each capability, f_{offer} is the funding counter offer received from the individual system and $finitial$ is the funding initially offered by the SoS manager. If the system is contributing to a critical capability for the SoS system, the SoS manager will allocate a higher reservation price for that individual system.

Negotiation protocol

The SoS manager initially sends a connectivity request to individual systems based on the selected SoS architecture. The connectivity requests include performance expectations for each capability, funding and deadline. Individual systems either accept or send a counter-offer to the SoS manager which may include changes in performance, funding or deadline. The SoS manager starts negotiation with the individual systems to find the optimal strategy for the SoS manager.

Finding the optimal strategy for the SoS manager:

Determining the optimal strategy for the SoS manager depends whether the SoS manager has full information on the acquisition environment or not. Two cases are considered to determine the optimal strategy for the SoS manager:

- Contracts under certainty: In this case, it is assumed that SoS manager can observe individual system's actions and there is no uncertainty about the individual system's actions. Thus the outcome is a function of the individual system's effort. In that case, SoS manager can use a forcing contract where the SoS manager pays the individual system only if it provides the outcome required by the SoS manager. In such situations, the contract should at least provide the individual system a reward where its utility will equal its reservation price. Thus the optimal reward will be determined based on the reservation price.

$$U^{is}\left(e^{*}, r(q^{*})\right) = \hat{u}$$

- Contracts under uncertainty: In this case, there is uncertainty about the possible outcomes of individual systems. Neither the SoS manager, nor the individual systems knows the certain state of the world when agreeing on the contract. Assuming that the SoS manager and individual systems are risk neutral, the maximization for the SoS utility function can be solved using a linear programming technique. The optimal reward for the individual systems will depend on the reservation price.

$$r_i = q_i - C$$

where C depends on the individual system rationality constraint.

3.4 Multi-Level optimization Model

3.4.1 Problem Formulation

In this section, we formulate the Stackelberg game between the SoS architect and the individual systems. The Acknowledged SoS manger referred here as architect is the leader of the Stackelberg game and decides on which capabilities should be provided by which systems and the funds allocated to the individual systems. It is assumed that the systems will provide the capabilities requested by the SoS architect. The systems are the followers and each individual system announces the deadline and performance level of the capabilities requested from it based on the funds allocated by the SoS architect. Each system is an individual decision maker and has no information about the SoS and other systems. In what follows, we first formulate the SoS architect's problem of building the SoS architecture. Then, the problem of an individual system is formulated considering the system's collaboration degree.

3.4.2 SoS Architect's Problem

Consider a System of Systems (SoS) that requires a set of capabilities. Let capabilities be indexed by i such that $i \in I$, $I = \{1, 2, \ldots, n\}$. The SoS architect can select among a set of systems to provide the capabilities. Let systems be indexed by j such that $j \in J$, $J = \{1, 2, \ldots, m\}$. Particularly, SoS architect can allocate multiple capabilities to a single system. A system, nevertheless, is not necessarily able to provide each capability. Let

$$a_{ij} = \begin{cases} 1 & \text{if capability } i \text{ can be provided by system } j; \\ 0 & \text{otherwise;} \end{cases}$$

and **A** be the $n \times m$-matrix of a_{ij} values. One set of decision variables of the SoS architect is then to select which systems will be asked to provide which capabilities. Let

$$x_{ij} = \begin{cases} 1 & \text{if capability } i \text{ is requested from system } j; \\ 0 & \text{otherwise;} \end{cases}$$

and **X** be the $n \times m$-matrix of x_{ij} values. Note that by definition of a_{ij}, we have $x_{ij} \le a_{ij}$. That is, the SoS architect will not request a capability from a system which cannot provide that capability. A system is selected in the SoS architecture if it is selected to provide at least one capability. Let

$$S_j = \begin{cases} 1 & \text{if } \sum_{i \in I} x_{ij} \ge 1; \\ 0 & \text{otherwise;} \end{cases}$$

that is, S_j is the binary (auxiliary decision) variable indicating selection of system j and let **S** be the m-vector of S_j values.

To function, the SoS should be fully connected. That is, each capability should have communication with every other capability. It is assumed that the capabilities provided by the same system are already connected. This suggests that the SoS architect should select an interface between any pair of selected systems. Note that $\frac{n(n-1)}{2}$ interfaces can be selected at most. In particular, let

$$y_{j_1 j_2} = \begin{cases} 1 & \text{if an interface is selected between systems } j_1 \text{ and } j_2; \; j_1; j_2 \in J; \\ 0 & \text{otherwise;} \end{cases}$$

and **Y** be the $m \times m$-matrix of $y_{j_1 j_2}$, $j_1, j_2 \in J$, values. Then, $y_{j_1 j_2} = 1$ if $S_{j_1} + S_{j_1} = 2$ and $I_{j_1 j_2} = 0$ if $S_{j_1} + S_{j_1} \le 1$. The following two points should be remarked: (i) $y_{jj} = 0$ $\forall j \in J$ as a system, by definition, can communicate with itself so no interface should be selected to enable communication of a system with itself, (ii) $y_{j_1 j_2} + y_{j_2 j_1} \le 1$ $\forall j_1, j_2 \in J$, that is, in the case systems j_1 and j_2 are selected it does not matter whether the SoS architect selects an interface between systems j_1 and j_2 or systems j_2 and j_1. These remarks, without loss of generality, can simply be handled by letting $y_{j_1 j_2} = 0$ if $j_1 \le j_2$.

In the process of SoS architecting, i.e., determining **X**, and thereby **S** and **Y**, the SoS architecture regards three objectives: minimization of the total cost of the SoS architecture, minimization of the deadline to complete the SoS architecture, and maximization of the total performance of the SoS architecture. Each interface has a fixed cost and let $h_{j_1 j_2}$ be the cost of selecting the interface between systems j_1 and j_2, $j_1, j_2 \in J$. Furthermore, we assume that each system requires a fixed cost for providing a specific capability. Let c_{ij} be the cost of requesting capability i from system j. Similarly, we assume that each system has different deadlines in providing capabilities and each system can provide different capability performance levels. The SoS architect, nevertheless, can allocate funds to the systems in order to improve the deadlines and performance levels of the capabilities requested from the systems. Let F_j denote the amount of funds allocated to system j and **F** be the m-vector of F_j values. Note that $F_j=0$ if $S_j=0$, i.e., no funds are given to a system that is not in the SoS architecture. The total cost of the SoS architecture then reads as

$$TC(\mathbf{X},\mathbf{S},\mathbf{Y},\mathbf{F}) = \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} + \sum_{j_1\in J}\sum_{j_2\in J} h_{j_1 j_2} y_{j_1 j_2} + \sum_{j\in J} F_j. \quad (1)$$

The first term of Equation (1) is the total cost of requesting capabilities from systems, the second term is the total cost of the interfaces selected, and the last term is the funds allocated to the systems.

The individual systems use the allocated funds to improve the deadlines and the performance levels of the capabilities requested from them. Later in this section, we will discuss how an individual system uses the allocated funds to improve the deadlines and the performance levels of the capabilities requested from it. Prior to that, our initial focus is on formulating the SoS architecture's problem. Let $d_{ij}(\mathbf{x}^j, F_j)$ and $p_{ij}(\mathbf{x}^j, F_j)$ denote the deadline and the performance level of system j in providing capability i when F_j amount of funds allocated to improve the performance measures of the requested capabilities, denoted by $\mathbf{x}^j$, where $\mathbf{x}^j$ is the j^{th} column vector of **X**. The deadline of the SoS architecture is defined as the maximum time required by a system in providing the capabilities requested from it. Particularly, $\max_{i\in I}\{d_{ij}(\mathbf{x}^j, F_j)x_{ij}\}$ is the deadline of system j. The deadline of the SoS architecture then reads as

$$DL(\mathbf{X},\mathbf{S},\mathbf{Y},\mathbf{F}) = \max_{j\in J}\{\max_{i\in I}\{d_{ij}(\mathbf{x}^j, F_j)x_{ij}\}\}. \quad (2)$$

The total performance of the SoS architecture is defined as follows

$$TP(\mathbf{X},\mathbf{S},\mathbf{Y},\mathbf{F}) = \sum_{i\in I}\sum_{j\in J} p_{ij}(\mathbf{x}^j, F_j)x_{ij}. \quad (3)$$

The SoS architect's goal is to create a SoS that provides each capability by at least one system while minimizing costs (Equation (1)) and deadline (Equation (2)) and maximizing the performance (Equation. (3)). The SoS architect's optimization problem then can be formulated as follows:

(SoS-A) : min $TC(\mathbf{X},\mathbf{S},\mathbf{Y},\mathbf{F})= \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij}+ \sum_{j_1\in J}\sum_{j_2\in J} h_{j_1j_2}y_{j_1j_2}+ \sum_{j\in J} F_j$

min $DL(\mathbf{X},\mathbf{S},\mathbf{Y},\mathbf{F})=\max_{j\in J}\{\max_{i\in I}\{d_{ij}(\mathbf{x}^j,F_j)x_{ij}\}\}$

max $TP(\mathbf{X},\mathbf{S},\mathbf{Y},\mathbf{F})= \sum_{i\in I}\sum_{j\in J} p_{ij}(\mathbf{x}^j,F_j)x_{ij}$

s.t. $\sum_{j\in J} a_{ij}x_{ij}\geq 1 \ \forall i\in I$ (4)

$S_j\leq \sum_{i\in I} x_{ij} \ \forall j\in J$ (5)

$S_j\geq \frac{1}{n}\sum_{i\in I} x_{ij} \ \forall j\in J$ (6)

$y_{jk}\geq S_j+S_k-1 \ \forall j\in J,\forall k\in J$ (7)

$x_{ij}\in\{0,1\} \quad \forall i\in I, \forall j\in J$ (8)

$S_j\in\{0,1\} \quad \forall j\in J$ (9)

$y_{jk}\in\{0,1\} \quad \forall j\in J,\forall k\in J$ (10)

$F_j\geq 0 \ \forall j\in J$ (11)

Equation (4) defines the constraints that each capability is requested from at least one system. Equations (5) and (6) guarantee that a system is selected in the SoS if at least one capability is requested from it and not selected otherwise. Particularly, if $\sum_{i\in I} x_{ij}=0$, Equation (5) indicates that $S_j=0$ as $S_j\in\{0,1\}$; and if $\sum_{i\in I} x_{ij}\geq 0$, $0<\frac{1}{n}\sum_{i\in I} x_{ij}\leq 1$; hence, Equation (6) indicates that $S_j=1$ as Equation (7) enforces the SoS architect to select the interfaces between any selected pair of systems. Note that if only one of the systems of any pair of the systems is selected, the interface between the systems in this pair will not be selected as there is a cost associated with it and interfaces do not contribute to the deadline and performance level of the SoS architecture. Equations (8), (9), and (10) define the binary restrictions on

the decision variables (including auxiliary decision variables) and Equation (11) is the non-negativity constraints for the funds allocated to the systems.

3.4.3 Systems' Problems

Given the allocated funds, F_j, and the capabilities requested from system j, $\mathbf{x}^j$, system j uses the allocated funds to improve performance measures of the requested capabilities. A system can improve the deadline and performance level of a requested capability. Particularly, system j can decrease the deadline of providing capability i by one time unit at a cost of d_{ij}. Similarly, system j can improve the performance level of capability i by one unit at a cost of p_{ij}. Let f_{ij}^1 and f_{ij}^2 denote the funds allocated by system j on decreasing the deadline and increasing the performance level of capability i, respectively. Note that $\sum_{i\in I} x_{ij}(f_{ij}^1+f_{ij}^2)\leq F_j$. Once the funds allocated to a system, they are not returned back to the SoS architect. We assume that, without any additional funds, d_{ij}^0 and p_{ij}^0 are the deadline and performance level of capability i by system j. That is, d_{ij}^0 is the maximum deadline and p_{ij}^0 is the minimum performance level announced by system j for providing capability i. Then, with the use of funds F_j, system j can have the following deadline and performance level for providing capability i:

$$d_{ij}(f_{ij}^1)=d_{ij}^0-d_{ij}f_{ij}^1 \qquad (12)$$

$$p_{ij}(f_{ij}^2)=p_{ij}^0+p_{ij}f_{ij}^2 \qquad (13)$$

It is assumed that there is a minimum deadline and maximum performance level achievable with funds in providing any capability by a system. Particularly, we let D_{ij} and P_{ij} denote the minimum deadline and maximum performance level achievable with funds in providing capability i by system j. That is, $d_{ij}(f_{ij}^1)\geq D_{ij}$ and $p_{ij}(f_{ij}^2)\leq P_{ij}$. It further indicates that $f_{ij}^1\leq \frac{d_{ij}^0-D_{ij}}{d_{ij}}$ and $f_{ij}^2\leq \frac{P_{ij}-p_{ij}^0}{p_{ij}}$.

In this setting, the system can use the funds allocated by the SoS architect towards two options: (i) improving the performance measures of the capabilities requested and (ii) creating surplus from the funds to increase its profitability. Option (i) would imply higher quality in providing capabilities; hence, it increases the chances of getting requests from the SoS architect. Option (ii), on the other hand, would imply lower quality in providing capabilities; hence, it decreases the chances of getting requests from the SoS architect. Therefore, there exists a trade-off between options (i) and (ii) for the system. In the case the system fully operates in favor of the SoS architect, it will allocate all of the funds provided by the SoS architect for improving the performance measures for the capabilities. In this case, we assume that the system's objective is to maximize the minimum percent improvement of the performance

measures. Specifically, percent improvement in deadline for providing capability *i* equals to $\frac{d_{ij}^0 - d_{ij}(f_{ij}^1)}{d_{ij}^0}$ and percent improvement in the performance level for providing capability *i* equals to $\frac{p_{ij}(f_{ij}^2) - p_{ij}^0}{p_{ij}^0}$. It then follows that a fully collaborative system would maximize $\min\left\{\frac{d_{ij}f_{ij}^1}{d_{ij}^0}, \frac{p_{ij}f_{ij}^2}{p_{ij}^0}\right\}$ if capability *i* is being provided.

However, a system does not need to be fully in favor of the SoS architect. A system would be more profitable if the percentage of the total funds used is lower. That is, the lower $\frac{F_j - \sum_{i \in I} x_{ij}(f_{ij}^1 + f_{ij}^2)}{F_j}$ is, the more benefits system *j* earns. To model an individual system's willingness to collaborate with the SoS architect, we define w_j as the collaboration degree of system *j*. Given w_j, system *j*'s objective function can be defined as follows:

$$SO_j(\mathbf{f}_j^1, \mathbf{f}_j^2) = w_j \sum_{i \in I} x_{ij} \left(\min\left\{ \frac{d_{ij}f_{ij}^1}{d_{ij}^0}, \frac{p_{ij}f_{ij}^2}{p_{ij}^0} \right\} \right) + (1 - w_j) \left(1 - \frac{1}{F_j} \sum_{i \in I} x_{ij}(f_{ij}^1 + f_{ij}^2) \right) \qquad (14)$$

where $\mathbf{f}_j^1$ and $\mathbf{f}_j^2$ are the *n*-vectors of f_{ij}^1 and f_{ij}^2 values, respectively. The first term of Equation 14 is the total minimum improvements of performance measures valued by the system and the second term is the total budget utilization in improving the performance measures valued by the system. In what follows, we present the mathematical formulation for system *j*'s problem:

$$(P^j): \quad \max \quad SO_j(\mathbf{f}_j^1, \mathbf{f}_j^2) = w_j \sum_{i \in I} \Delta_{ij} + (1 - w_j) \left(1 - \frac{1}{F_j} \sum_{i \in I} x_{ij}(f_{ij}^1 + f_{ij}^2) \right)$$

$$\text{s.t.} \quad \sum_{i \in I} x_{ij}(f_{ij}^1 + f_{ij}^2) \le F_j \qquad (15)$$

$$\Delta_{ij} \le x_{ij} \frac{d_{ij}f_{ij}^1}{d_{ij}^0} \; \forall i \in I \qquad (16)$$

$$\Delta_{ij} \leq x_{ij} \frac{p_{ij} f_{ij}^2}{p_{ij}^0} \quad \forall i \in I \tag{17}$$

$$f_{ij}^1 \leq \frac{d_{ij}^0 - D_{ij}}{d_{ij}} \quad \forall i \in I \tag{18}$$

$$f_{ij}^2 \leq \frac{P_{ij} - p_{ij}^0}{p_{ij}} \quad \forall i \in I \tag{19}$$

$$f_{ij}^1 \geq 0 \quad \forall i \in I \tag{20}$$

$$f_{ij}^2 \geq 0 \quad \forall i \in I \tag{21}$$

Equation (15) guarantees that system *j* will not allocate funds in improvements more than the funds allocated by the SoS architect. Equations (16) and (17) define $\Delta_{ij} = \min\left\{ \frac{d_{ij} f_{ij}^1}{d_{ij}^0}, \frac{p_{ij} f_{ij}^2}{p_{ij}^0} \right\}$ for requested capability *i*. . . Equations (18) and (19) state the upper bounds on the funds that can be allocated to improve a performance measure. Equations (20) and (21) are the non-negativity of f_{ij}^1 and f_{ij}^2, respectively.

3.4.4 Stackelberg Formulation

It is well known that Stackelberg games can be formulated as bi-level optimization problems. In what follows, we therefore present the bi-level model of the Stackelberg game between the SoS architect and the systems. Particularly, let $\mathbf{f}_j^{1*}(\mathbf{x}^j, F_j)$ and $\mathbf{f}_j^{2*}(\mathbf{x}^j, F_j)$ be the solution of P^j given $\mathbf{x}^j$ and F_j. Then,

$$d_{ij}(\mathbf{x}^j, F_j) = d_{ij}^0 - d_{ij} f_{ij}^{1*}(\mathbf{x}^j, F_j), \tag{22}$$

$$p_{ij}(\mathbf{x}^j, F_j) = p_{ij}^0 + p_{ij} f_{ij}^{2*}(\mathbf{x}^j, F_j). \tag{23}$$

In the bi-level model, the upper level is the SoS architect's problem, i.e., SoS-A. There are *m* lower level problems, consisting of the systems' problems, i.e., $P^j \ \forall j \in J$. The following bi-level model represents the SoS architecturing with individual system contracts (SoS-ISC):

(SoS-ISC) : min $TC(\mathbf{X},\mathbf{S},\mathbf{Y},\mathbf{F})$

min $DL(\mathbf{X},\mathbf{S},\mathbf{Y},\mathbf{F})$

max $TP(\mathbf{X},\mathbf{S},\mathbf{Y},\mathbf{F})$

s.t. Equations (4)-(11)

Equations (22)-(23)

$(\mathbf{f}_j^{1*}(\mathbf{x}^j,F_j),\mathbf{f}_j^{2*}(\mathbf{x}^j,F_j))=\text{argmax}\{SO_j(\mathbf{f}_j^1,\mathbf{f}_j^2)\ s.t.\ Equations\ (15)-(21)\}\ \ \forall j\in J.$

SoS-ISC is a multi-objective multi-level optimization problem. In the next section, we discuss a genetic algorithms based meta heuristic to solve SoS-ISC.

3.4.5 Solution Analysis

Our focus is to determine a set of (**X**,**S**,**Y**,**F**), i.e., SoS architect with individual system contracts. Recall that once **X** is known, **S** and **Y** can easily be determined. Therefore, we propose a two-stage approach to solve SoS-ISC. First, assuming that **X** is given, we determine **F** with a local search heuristic. That is, given the requested capabilities from the systems, a heuristic method to build individual system contracts is constructed. We refer to this stage as the *system contracting*. This enables us to evaluate the given **X**. Then, a genetic algorithm is structured for determining a set of **X** solutions. We refer to this stage as *capability assignment*. An illustration of this meta-heuristic is given in Figure 10.

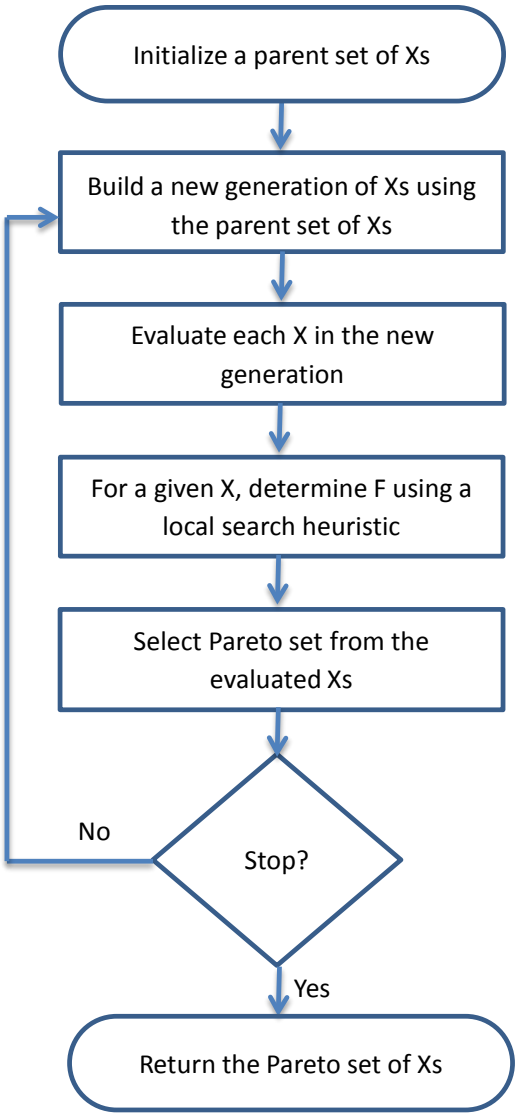

Figure 10. Flow Chart of the Meta-Heuristic Approach

3.4.6 System Contracting

Here, we assume that **X** is given, that is, the SoS architect has decided on which capabilities are requested from which systems. Then, the SoS architect aims to allocate funds to each individual system, from which at least one capability is requested. We assume that the SoS architect will try to establish a contract with each such system in the SoS individually. Therefore, we investigate a heuristic approach to determine $F_j \; \forall j$ such that S_j=1 under the given **X**.

Now, without loss of generality, suppose that capabilities 1, 2,…,*k* such that $k \le n$ are requested from system *j*. The SoS architect desires to get the most performance improvement with the funds allocated to system *j*. Therefore, we assume that, in determining F_j the SoS architect aims to maximize the total improvement per money unit spent. Recall that Δ_{ij} is defined as the minimum improvement in capability *i* as a result of the system *j*'s decision on how to use funds, F_j, allocated to it. This then suggests that the bigger the $\frac{1}{F_j}\sum_{i=1}^{k}\Delta_{ij}$ the more benefits the SoS architect received.

Furthermore, recall that $0 \le f_{ij}^1 \le \frac{d_{ij}^0 - D_{ij}}{d_{ij}}$ and $0 \le f_{ij}^2 \le \frac{P_{ij} - p_{ij}^0}{p_{ij}}$. That is, the money units that system *j* can spend on improvement of any performance measure of any requested capability has upper and lower bounds. At this point, in our heuristic approach, we assume that the SoS architect will determine F_j as follows: either $\lambda_{ij}^1 = 0$ or $\lambda_{ij}^2 = \frac{d_{ij}^0 - D_{ij}}{2d_{ij}}$ or $\lambda_{ij}^3 = \frac{d_{ij}^0 - D_{ij}}{d_{ij}}$ will be allocated for improving the deadline of capability $i \le k$ requested from system *j*; and, $\beta_{ij}^1 = 0$ or $\beta_{ij}^2 = \frac{P_{ij} - p_{ij}^0}{2p_{ij}}$ or $\beta_{ij}^3 = \frac{P_{ij} - p_{ij}^0}{p_{ij}}$ will be allocated for improving the performance level of capability $i \le k$ requested from system *j*. Given the selected funds anticipated by the SoS architect to be allocated by system *j* in improving deadline and performance level of the capabilities, F_j will be sum of these funds. Then, P^j can be solved to calculate the actual improvements with the total funds allocated to system *j*. After that, the SoS can evaluate the $\frac{1}{F_j}\sum_{i=1}^{k}\Delta_{ij}$ value.

In selecting which level of funds, λ_{ij}^1 or λ_{ij}^2 or λ_{ij}^3 and β_{ij}^1 or β_{ij}^2 or β_{ij}^3, we propose a local search heuristic. Particularly, let

$$\text{»}_{ij} = \begin{cases} 1 & \text{»}_{ij}^1 \text{ is selected;} \\ 2 & \text{»}_{ij}^2 \text{ is selected;} \\ 3 & \text{»}_{ij}^3 \text{ is selected;} \end{cases}$$

and let $\boldsymbol{\lambda}_j$ be the *k*-vector of λ_{ij} values. Similarly, let

$$^2_{ij} = \begin{cases} 1 & ^2_{ij}^1 \text{ is selected;} \\ 2 & ^2_{ij}^2 \text{ is selected;} \\ 3 & ^2_{ij}^3 \text{ is selected;} \end{cases}$$

and let $\boldsymbol{\beta}_j$ be the *k*-vector of β_{ij} values. Note that given $\boldsymbol{\lambda}_j$ and $\boldsymbol{\beta}_j$, F_j can be calculated easily. The following local search heuristic determines the funds allocated to system *j* by the SoS architect.

Algorithm 1 System Contracting with Local Search Heuristic for System j (SC-LHS^j):

Step 0: Without loss of generality, order the capabilities requested from system j such that capabilities 1,2,…,k are requested. Initialize $»_{ij}=^2{}_{ij}=2$ ∀i,i=1,2,…,k. Let $È=[»_j,^2{}_j]$ and $È^*=È$.

Step 1: Find the best contract with fund cutting, $\psi^{[-1]}$

Step 1.1. For ℓ=1:2k

Step 1.2. Set $\psi_\ell^{[-1]}=\psi$ and if $\psi_\ell^{[-1]}>1$, set $\psi_\ell^{[-1]}=\psi_\ell^{[-1]}-1$

Step 1.3. Solve P^j under with $F_j=F^\ell=\sum_{\ell=1}^{2k}\psi_\ell^{[-1]}$

Step 1.4. Calculate $\tau_\ell(\psi_\ell^{[-1]})=\frac{1}{F^\ell}\sum_{i=1}^{k}\Delta_{ij}$

Step 1.5. End

Step 1.6. $\psi^{[-1]}=\text{argmin}\{\tau_\ell(\psi_\ell^{[-1]})\}$

Step 2: Find the best contract with fund raising, $\psi^{[+1]}$

Step 2.1. For ℓ=1:2k

Step 2.2. Set $\psi_\ell^{[+1]}=\psi$ and if $\psi_\ell^{[-1]}<2$, set $\psi_\ell^{+1]}=\psi_\ell^{[+1]}+1$

Step 2.3. Solve P^j under with $F_j=F^\ell=\sum_{\ell=1}^{2k}\psi_\ell^{[+1]}$

Step 2.4. Calculate $\tau_\ell(\psi_\ell^{[+1]})=\frac{1}{F^\ell}\sum_{i=1}^{k}\Delta_{ij}$

Step 2.5. End

Step 2.6. $\psi^{[+1]}=\text{argmin}\{\tau_{\ell}(\psi_{\ell}^{[+1]})\}$

Step 3: Let $\overline{\psi}=\text{argmin}\{\tau(\psi^{[-1]}),\tau(\psi),\tau(\psi^{[+1]})\}$. If $\overline{\psi}=\psi^{*}$, stop and return ψ^{*}. Else, let $\psi=\overline{\psi}$ and go to Step 1.

3.4.7 Capability Assignment

In this section, we explain the details of the genetic algorithm (GA) proposed to solve SoS-ISC. The GA consists of four main steps: chromosome representation, fitness evaluation, mutation, and termination.

Chromosome Representation and Initialization:

As introduced in Section 3.2.1, We adopt the binary matrix representation of **X** as the chromosome. Specifically, note that $\sum_{j=1}^{m} a_{ij}x_{ij}=1$ in a feasible **X**. Therefore, for each capability *i*, we select a system *j* among the systems with $a_{ij}=1$ randomly and make x_{ij} value equal. Repeating this process for each capability, a feasible **X** is generated. There are two advantages of this chromosome representation: (i) feasibility of each chromosome is guaranteed and (ii) mutation operations, as will be explained, are really simple to generate new chromosomes. We set the initial population size equal to *N*.

Fitness Evaluation:

Recall that SoS architect's problem is a multi-objective optimization problem. A common approach to solve multi-objective optimization problems is to generate a set of Pareto efficient solutions. For the problem of interest, an SoS architecture with individual system contracts, i.e., say ($\mathbf{X}^0$,$\mathbf{S}^0$,$\mathbf{Y}^0$,$\mathbf{F}^0$) is Pareto efficient if there is not other (**X**,**S**,**Y**,**F**) that is better than ($\mathbf{X}^0$,$\mathbf{S}^0$,$\mathbf{Y}^0$,$\mathbf{F}^0$) in terms of all of the three objective functions defined in Equations (1), (2) and (3). That is, ($\mathbf{X}^0$,$\mathbf{S}^0$,$\mathbf{Y}^0$,$\mathbf{F}^0$) is Pareto efficient if it is not dominated by any other solution (**X**,**S**,**Y**,**F**). Therefore, given a set of chromosomes, instead of finding a best one, we focus on determining the Pareto efficient chromosomes in the current population. Let PF^{pop} denote the set of Pareto efficient solutions in the set of chromosomes of the current population, denoted by *POP*. The following algorithm generates PF^{pop}.

Algorithm 2 Determining Pareto efficient solutions of a given population:

Step 0: Let TC_l, DL_l, and TP_l denote the objective function values of l^{th} chromosome, $chrom^l$:

Step 1: For l=1:|POP|

Step 2: For k=l+1:|POP|

Step 3: If $TC_l<TC_k$, $DL_l<DL_k$, and $TP_l>TP_k$

Step 4: Set POP:=POP−{$chrom^k$}

Step 5: Else, if $TC_l>TC_k$, $DL_l>DL_k$, and $TP_l<TP_k$

Step 6: Set POP:=POP−{$chrom^l$}

Step 7: End

Step 8: End

Step 9: Return PF^{pop}=POP.

We use PF^{pop} as the parent of the next population to be generated.

Mutation:

Due to the fact that one should have $\sum_{j=1}^{m} a_{ij}x_{ij}=1$ for each capability, a new chromosome can be generated by changing the system a specific capability is requested from. Note that $\sum_{j=1}^{m} a_{ij}$ is the number of systems that can provide capability i and the number of feasible **X** chromosomes is then equal to $\prod_{i=1}^{n}\left(\sum_{j=1}^{m} a_{ij}\right)$. With a single change in the provider system for each capability, at most 2^n new **X** chromosomes can be generated from a given **X** chromosome. We prefer to generate 2^k new chromosomes from each parent chromosome such that $k\leq n$. To do so, k capabilities are randomly selected. Without loss of generality, suppose that capabilities 1,2,…,k are selected. Then, for each selected capability, we randomly select a system such that $a_{ij}=1$ and $x_{ij}\neq 1$. This generates a new chromosome. After that, the same process applied to the other selected capabilities. Therefore, at the end of mutations, we generate at most $2^k|PF^{pop-1}|$, where PF^{pop-1} is the set of parent chromosomes. Then $|POP|=(2^k+1)|PF^{pop-1}|$. It is also important to note that the new generation consists of the newly

generated chromosomes plus the parent chromosomes. That is, we do not omit the parent chromosomes from the new generation as they can still be Pareto efficient compared to the newly generated chromosomes. In fitness evaluation step, this guarantees that the set of the parent chromosomes for the next generations are not worsening.

Termination:

As a termination criteria, we consider the parent chromosomes at the end of fitness evaluation of each generation. Particularly, if $PF^{pop}=PF^{pop+1}$, it means that the later generation did not result in new Pareto efficient chromosomes. If $PF^{pop}=PF^{pop+1}$ for ℓ consecutive generations, the algorithm stops. The parent chromosomes at termination constitute the set of Pareto efficient solutions provided to the SoS architect.

3.5 Negotiation Between SoS and System Providing Capability to SoS

This section of the report describes the individual system negotiation models in detail. Each negotiation model is based on three major issues, which include performance, funding and deadline. The SoS agent provides each system with a subsection of the whole genetic string (or meta-architecture). The sectioned string is a matrix. SoS agent gives the required performance levels, funding provided and deadlines for each capability, the system can provide. For example System 1 can provide capabilities 1 and 5, so the matrix will include 2 rows of information on the 3 issues for each capability. Besides, the matrix also includes the interfaces System 1 has to make with other systems, as required by the SoS manager. Therefore, in a way each individual system gets a piece of the full chromosome and in turn partial information. The inputs from the SoS listed above are saved in an n by (m+4) matrix shown.

	Architecture (the portion related to system j)																							deadline	funding	performance
	S_j	$I_{j,1}$	$I_{j,2}$	$I_{j,3}$	$I_{j,4}$	$I_{j,5}$	$I_{j,6}$	$I_{j,7}$	$I_{j,8}$	$I_{j,9}$	$I_{j,10}$	$I_{j,11}$	$I_{j,12}$	$I_{j,13}$	$I_{j,14}$	$I_{j,15}$	$I_{j,16}$	$I_{j,17}$	$I_{j,18}$	$I_{j,19}$	$I_{j,20}$	$I_{j,21}$	$I_{j,22}$	$SoS.d_j$	$SoS.f_j$	$SoS.p_j$
C_1																										
C_2																										
C_3																										
C_4																										
C_5																										

Figure 11: Inputs from SoS to System

The systems respond to the SoS offer in a similar format. The system agents calculate the difference in values for each attribute calculated based on their utility and the SoS offer. These values are called the delta values are chosen as a response of the systems to SoS in the matrix format.

At first, the individual system will receive the requirements from SoS. The target of SoS is to obtain n capabilities by m systems and some connections between these systems. The input information from SoS will include deadline, funding and performance demands for each capability, and those interface need to be construct by the *i*th system with others. Then if SoS has m systems and needs n capabilities, the input is an n by m+4 matrix as illustrated in Table 4 below.

Table 4. SoS output format

		Interface between Sj and Si				deadline	funding	performance
	Sj	Ij,1	Ij,2	….	Ij,m	SoS.di	SoS.fi	SoS.pi
C1	1	0	1	…	1	1	1	5
C2	0	0	0	…	0	0	0	0
…	…	…	…	…	…			
Cn	1	1	0	…	0	3	2	2

One complete epoch consists of an offer by the SoS agent to the system agents and a counter offer in reply from the systems to the initiator. The algorithm below outlines the procedure of negotiation protocol.

Simplified negotiation protocol algorithm

```
epoch = 0
while none of the agents have conceded
    SoS makes an offer to the System Agent
    System Agent replies with an offer to the SoS
    epoch = epoch + 1
return offer
```

Each individual system may possess more than one capability. The SoS manager may request to procure these capabilities at different levels of performance and would provide different funding amount for each of them. Another assumption is that system agents have fixed amount of resources and work equally well to develop all capabilities asked of them. Now the resources have to be shared amongst the development of capabilities in a certain ratio of agent preference. The agent ranks the priority of development of a capability based on information provided by the SoS and shares the resources accordingly to develop them. In this model, both the SoS agent and individual system agents assume partial information.

3.5.1 Selfish System Model

Model Overview:

A model of an individual system *k* is built, which is capable of providing both capabilities to a system of systems (SoS) and interfaces with other individual systems in the SoS. The request for participation is sent from SoS to the individual system, including:

- Requested capabilities, C_i
- Requested interfaces between the system *k* and other individual system *j* on capability *i*, χ_{ij}
- Deadline of delivery, $SoS.d_i$
- Funding for providing the requested capabilities, $SoS.f_i$

- The performance requirement on each of requested capabilities, $SoS.p_i$

This model can be termed a "selfish" model in that the necessary condition for the individual system *k* to collaborate with the SoS is that the incremental profit from the participation is nonnegative. Therefore, a resource allocation problem is formulated to model the decision behavior of the individual system. The optimization problem is solved with considerations of the capabilities, resources and efficiency of the system. Moreover, the market condition is modeled so that the system agent has a rational assessment of the incremental profit provided by the SoS.

The outputs sent from the system k to the SoS include:

- Provided capabilities, C_i
- Provided interfaces between the system *k* and other individual system *j* on capability *i*, χ_{ij}
- Delivery time deviation (additional time needed), $SoS.\Delta d_i$
- Funding deviation (additional fund needed), $SoS.\Delta f_i$
- Performance deviation (under performance is any), $SoS.\Delta p_i$

The outputs listed above are saved in an n by (m+4) matrix shown.

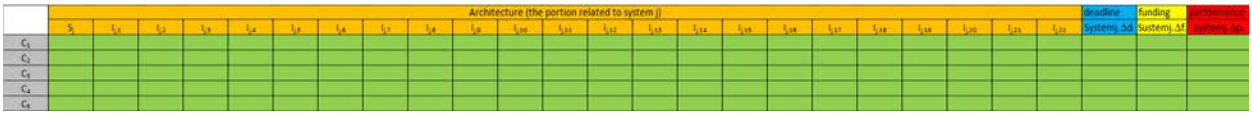

Figure 12: Outputs from System to SoS

Model Description:

Two linear programming (LP) models are built to provide two scenarios of negotiations with the SoS, which are solved using the optimization tool in Matlab. In the first scenario, the SoS is informed possible performance deviation (if any) at the provided funding and time. In the second scenario, the SoS may be provided the capabilities and interfaces as it desires, yet it may be asked to provide additional funding and/or time.Model setting

The following is the setting of the individual system *k*.

i: the index of capabilities, and $i = 1, 2, \ldots, n$.

I: The set of capability indices, and $I = \{1, 2, \ldots, n\}$.

j: the index of individual systems, and $j = 1, 2, \ldots, m$.

J: The set of system indices and $J = \{1, 2, \ldots, m\}$.

C_i: binary variable indicating whether the capability *i* is requested, $\forall$ *i.*

I_{req}: the set of requested capability indices; $I_{req} \subseteq I$.

p_i: performance requirement on capability i, $\forall\ i \in I_{req}$.

d_i: deadline to deliver capability i, $\forall\ i \in I_{req}$.

f_i: funding provided to capability i, $\forall\ i \in I_{req}$.

χ_{ji}: binary variable indicating the requested interface between system j and k ($k \neq j$) in forming capability i, $\forall\ i$ and j.

n_i: the number of interfaces that system k is requested to provide on capability i, $\forall\ i \in I_{req}$.

y_i: the system k's throughput of capability i in a unit of time, $\forall\ i \in I$.

I_y: the set of indices of capabilities that system k is able to build, $I_y = \{i \mid y_i > 0\}$.

I_{reqf}: the set of indices of capabilities that is requested by the SoS and the system k is able to build, and $I_{reqfsbl} = I_y \cap I_{req}$.

$I_{reqifsbl}$: the set of indices of capabilities that is requested by the SoS but the system k is not able to build, and $I_{reqifsbl} = \bar{I}_y \cap I_{req}$.

Z_{avg}: the system k's average available resource per unit of time.

Z_{range}: the range of the system k's resource per unit of time.

T: planning time horizon. $T = \max\{\text{int}\lceil \sum_{i \in Ireq} p_i/(y_i \times Z_{min}) \rceil, \max_{i \in Ireq}(d_i)+1\}$. $\text{int}\lceil \sum_{i \in Ireq} p_i/(y_i \times Z_{min}) \rceil$ is the time needed (in integer) if the system k has only the minimum resource and build the requested capabilities in sequence. $[\max_{i \in Ireq}(d_i)]$ is the relaxed upper bound of deadlines. The model assumes the planning time horizon is no shorter than these two.

t: the index of time; $t = 1, 2, \ldots, T$.

Z_t: the system k's resource available at time t, and $Z_{min} = Z_{avg} - 0.5Z_{range} \leq Z_t \leq Z_{max} = Z_{avg} + 0.5Z_{range}$, $\forall\ t$.

Z_{ti}: the resource allocated to build capability i at time t. $z_{it} \geq 0\ \forall\ i \in I_{reqfsbl}$ and $\forall\ t$. $z_{it\ is}$ are the decision variables.

c_{ci}: the cost of consuming one unit of resource in providing capability i at time zero, $\forall\ i \in I_y$.

c_{li}: the cost of consuming one unit of resource in providing interfaces associated with capability i, as a percentage of c_{ci}.

g: the inflation rate of unit cost over time.

c'_{ti}: the cost of consuming one unit of resource in providing capability i and the associated interfaces at time t, for $t = 0, 1,\ldots,.d_i$ and $i \in I_y$. $c_{ti} = c_{ci}(1+c_{li}n_i)\exp(g(t-0))$.

pm: profit margin.

c_{ti}: the market price of one unit of resource for providing capability i at time t. $c_{ti} = (1+\text{pm})c'_{ti}$.

c''_{ti} (>0): virtual penalty on exceeding the deadlines, for $d_i < t \leq T$ and $i \in I_y$.

cp_{ti} (= $c'_{ti} + (c_{ti} - c'_{ti})^+ + c''_{ti}$): the cost of consuming one unit of resource for providing capability i and the associated interfaces at time t, after the adjustment for the opportunity cost, $\forall\ t$.

w_i : the weight that system k put on capability i, $\forall\ i \in I_{reqfsbl}$; and $w_i = [f_i/d_i]/\ (\sum_{i \in Ireq} f_i/d_i)$.

Linear Programming (LP) models for decision support:

The first problem (P1) helps identify the best performance of system k on the requested capabilities and interfaces with the given funding and deadlines. First, $z_{ti} = 0$ for $i \notin (I_{reqfsbl})$; and $\forall i \in I_{reqfsbl,}$ z_{ti}'s are determined by the following problem:

$$\min \sum_{i \in I_{reqfsbl}} \left(-w_i \sum_{t=1}^{d_i} y_i z_{ti} \right)$$

$$\text{subject to:}$$

$$\sum_{t=1}^{d_i} y_i z_{ti} \leq p_i, \forall i \in I_{reqfsbl};$$

$$\sum_{i \in I_{reqfsbl}} z_{ti} \leq Z_t, \text{for } t = 1,2,\cdots, \max_{i \in I_{reqfsbl}} \{d_i\};$$

$$\sum_{i \in I_{reqfsbl}} \sum_{t=1}^{d_i} c_{ti} z_{ti} \leq \sum_{i=1}^{n} f_i;$$

$$z_{ti} \geq 0, \forall i \in I_{reqfsbl} \text{ and for } t = 1,2,\cdots,d_i. \qquad (3.6.1\text{-}1)$$

The objective of (P1) is to minimize the weighted sum of under performances. The first constraint means the performance on a requested capability does not exceed the required performance. The second constraint means the resource consumed at time t should not exceed the resource available at then. The third constraint means the total costs should not exceed the total funding provided. The fourth constraint indicates that the resource relocated to building a requested capability and the associated interfaces is nonnegative.

Denote by $\{z_{ti}^* \mid t = 1, 2, \ldots, d_i, \text{ and } i \in I_{reqfsbl}\}$ an solution to (P1). Let $z_{ti}^* = 0$ at $t = 1, 2, \ldots, d_i$ and for $i \notin I_{reqfsbl}$ so that z_{it}^* is defined at any i. The minimized performance deviation is calculated by

$$\Delta p_i^* = y_i \sum_{t=1}^{d_i} z_{ti}^* - p_i \quad . \quad (3.6.1\text{-}2)$$

The minimized performance deviation is calculated based on the assumptions of no extension of deadlines and/or no additional funding:

$$\Delta d_i^* = 0 , \quad (3.6.1\text{-}3)$$

$$\Delta f_i^* = 0 . \quad (3.6.1\text{-}4)$$

The second problem (P2) helps determine the minimum additional funding and/or minimum additional time to meet the goal of forming all the capabilities and interfaces that system *k* is capable of providing. First, $z_{ti} = 0$ for $I \notin I_{reqfsbl}$; and $\forall\ I \in I_{reqfsbl}$, z_{ti}'s are determined by the following problem:

(P2)

$$\min \left\{ \sum_{i \in I_{reqfsbl}} \sum_{t=1}^{T} cp_{ti} z_{ti} \right\}$$

subject to:

$$y_i \sum_{t=1}^{T} z_{ti} = p_i, \forall i \in I_{reqfsbl}$$

$$\sum_{i \in I_{reqfsbl}} z_{ti} \le Z_t, \forall t$$

$$z_{ti} \ge 0, \forall t \text{ and } i \in I_{reqfsbl} \quad (3.6.1\text{-}5)$$

The objective of (P2) is to minimize the total costs, including the additional fund used and the virtual penalty on additional time used. The first constraint means the performance on a requested capability is equal to the required performance. The second constraint means the resource consumed at time *t* should not exceed the resource available at then. The third constraint indicates that the resource relocated to building a capability and the associated interfaces is nonnegative.

In (P2), the funding constraint is relaxed and the deadline for finishing any capability *i* is "extended" to *T*. We choose *T* to be

$$T = \max \left\{ \text{int} \left\{ \sum_{i \in I_{reqfsbl}} \left\lceil \frac{p_i}{Z_{\min} y_i} \right\rceil \right\}, \max_{i \in I_{reqfsbl}} (d_i) \right\} , \quad (3.6.1\text{-}6)$$

$\text{int}\left\{\sum_{i\in I_{reqfsbl}}\left\lceil\frac{p_i}{Z_{\min}y_i}\right\rceil\right\}$ is the time needed if building capabilities in a sequential manner and the system *k* has just the minimum resource. The SoS may provide a very long deadline that is not needed by the system k. Considering this possibility, we choose such a value of *T* so that a feasible solution to (P2) is assured.

$\{c''_{ti}|t = d_i + 1, \ldots, T;\ i \in I_{reqfsbl}\}$ is the virtual penalty on exceeding the deadlines. For any capability *i*, the penalty satisfies

$$c''_{(d_i+1)i} \geq \max_{1\leq t\leq d_i} c_{it}, \qquad (3.6.1\text{-}7)$$

and

$$c''_{ti} > c''_{si},\ \forall t > s > d_i. \qquad (3.6.1\text{-}8)$$

The virtual penalty like such indicates that the cost to build any capability *i* after the desired deadline, d_i, is extremely high and the marginal cost grows with the prolonged time. Consequently, the objective function of (P2) discourages the use of the resource after the deadlines unless system *k* has to. Therefore, the objective function of (P2) effectively penalizes the usages of both the extra funding and time.

Denote by $\{\hat{z}_{ti}|\ t = 1, 2, \ldots, T;\ i \in I_{reqfsbl}\}$ an solution to (P2). Let $\hat{z}_{ti}$=0 at any time *t* and for $i \notin I_{reqfsbl}$ to make $\hat{z}_{ti}$ be defined at any *i*. Since the system *k* guarantees the performance on the capabilities that it is able to provide, the minimum deviation of performance is either zero or $-p_i$. The minimum additional time needed is found to be

$$\Delta d_i^* = \begin{cases} \hat{d}_i - d_i & i \in I_{reqfsbl} \\ \inf & i \in I_{reqifsbl} \\ 0 & i \in \overline{I}_{req} \end{cases} \qquad (3.6.1\text{-}9)$$

where

$$\hat{d}_i = \max_{1\leq t\leq T}\{t \mid \hat{z}_{ti} > 0\},\ \forall\ i \in I_{reqfsbl}. \qquad (3.6.1\text{-}10)$$

The total minimum additional funding is

$$\Delta f^* = \left(\sum_{i\in I_{reqfsbl}}\sum_{t=1}^{d_i} c_{ti}\hat{z}_{ti} - \sum_{i=1}^{n} f_i\right)^+, \qquad (3.6.1\text{-}11)$$

which is further split as the additional funding needed for each requested capability:

$$\Delta f_i^* = \lambda_i \Delta f^* \quad \text{(3.6.1-12)}$$

Where λ_i is determined by

$$\lambda_i = \begin{cases} \left(\sum_{t=1}^{\hat{d}_i} c_{ti}^* z_{ti} - f_i \right)^+ \Big/ \sum_{i \in I_{reqfsbl}} \left(\sum_{t=1}^{\hat{d}_i} c_{ti}^* z_{ti} - f_i \right)^+ & i \in I_{reqfsbl} \\ \text{Inf} & i \in I_{reqifsbl} \\ 0 & i \in \overline{I}_{req} \end{cases} . \quad \text{(3.6.1-13)}$$

Negotiation with the SoS:

If the system k is not able to provide any capability requested by the SoS, that is, $I_{reqfsbl} = \varnothing$, the system k will tell the SoS that the performance deviation is zero, but the deviations of deadline and funding is extremely large:

$$System_k.\Delta p_i = 0, \ \forall i \quad \text{(3.6.1-14)}$$

$$System_k.\Delta d_i = \inf, \ \forall i \quad \text{(3.6.1-15)}$$

$$System_k.\Delta f_i = \inf, \ \forall i \quad \text{(3.6.1-16)}$$

Otherwise, the solutions to (P1) and (P2) generate the foundation of two negotiation scenarios for the system *k*. We assume these two scenarios occur with chances. The system *k* may not share the complete information (e.g., the best performance, the capabilities not capable of providing, cost information) with the SoS during the negotiation for some reasons (e.g., business secret, for better negotiation outcomes).

The first scenario of negotiation is derived from the solution to (P1). The agent of system *k* provides the SoS the performance deviation defined as:

$$System_k.\Delta p_i = \begin{cases} \Delta p_i^* & i \in I_{req} \\ 0 & \text{otherwise} \end{cases} , \quad \text{(3.6.1-17)}$$

with the given funding and time:

$$System_k.\Delta d_i = \left[0, 0, \cdots, 0 \right]^T , \quad \text{(3.6.1-18)}$$

$$System_k.\Delta f_i = \left[0,0,\cdots,0\right]^T \quad . \qquad (3.6.1\text{-}19)$$

Δp_i^* in Eqn. (3.6.1-17) has been defined in (3.6.1-2). In this scenario of negotiation, the system *k* has strong motivation to participate and, therefore, its shows the minimum deviation of performance to SoS, as it is indicated by Eqn. (3.6.1-17). Since the system *k* does not want to let the SoS know what capabilities it is not capable of providing, the system *k* does not update the C_i's and χ_{ji}'s.

The second scenario of negotiation is derived from the solution to (P2). The agent of system *k* provides the SoS the performance deviation defined as:

$$System_k.\Delta p_i = \begin{cases} 0 & i \in I_{reqfsbl} \\ -p_i & \text{otherwise} \end{cases} , \qquad (3.6.1\text{-}20)$$

that is the system k fully meets the performance requirement on the capabilities that it is capable of providing. The system agent adds a random nonnegative number above the minimum additional fund needed in order to not share the private information with the SoS. If the additional fund needed is zero, the agent will still ask for an additional funding equal to a small portion of the provided funding. On those capabilities that the system k is not capable of providing, the agent asks for a very large amount of additional funding equal to Mf_i. Therefore,

$$Systemk.\Delta f_i = \begin{cases} \Delta f_i^*(1+\theta_i) & i \in I_{reqfsbl} \text{ and if } \Delta f_i^* \neq 0 \\ \theta_i f_i & i \in I_{reqfsbl} \text{ and if } \Delta f_i^* = 0 \\ Mf_i & i \in I_{reqifsbl} \\ 0 & i \in \bar{I}_{req} \end{cases} . \qquad (3.6.1\text{-}21)$$

The agent adds a random nonnegative integer above the minimum time in order to not share the private information with the SoS. If the system k is not capable of providing a capability, then the agent asks for a very long additional time,

$$\Delta d_i = \begin{cases} \Delta d_i^* + \varepsilon_i & i \in I_{reqfsbl} \\ Md_i & i \in I_{reqifsbl} \\ 0 & i \in \bar{I}_{req} \end{cases}$$

(3.6.1-22)

Again, since the system *k* does not want to let the SoS know what capabilities it is not capable of providing, the system *k* does not update the C_i's and χ_{ji}'s.

Variation of the Individual System:

Technically, we can produce a family of models by modifying the model parameters. An excel file (setting_systemk.xlsx) is created as an input file for the matlab code of this model, which lists the model parameters that can be modified to create a family of models with different characteristics and behaviors. In the following these parameters, and the way of selecting values for these parameters, are discussed in sequence.

- y: throughput (units of capability produced per time unit per resource unit). It is an nonnegative real value vector of *n* elements. "y(i)=0" means the individual system k is not capable of providing capability i. The greater the y(i), the more efficient the system k in building the capability i.
- Resource per time unit (assuming a symmetric distribution of resource)
    - Z_avg: a positive real number. The greater the Z_avg, the higher the average resource of the system k.
    - Z_range: X_range is a positive real number and no greater than 200% of Z_avg. The greater the Z_avg, the more volatile the resource of system k.
- Costs
    - c_c: the cost of consuming one unit of resource per time unit in producing capabilities. It is a nonnegative vector of n elements. "c_c(i)= 0" if "y(i)=0". The greater the c_c(i), the more expensive for the system k to provide capability i.
    - c_I: the cost of consuming one unit of resource per time unit in producing interfaces, as a percentage of c_c. It is a vector of n elements. "c_I(i) = 0" if "y(i)=0". The greater the c_I(i), the more expensive for the system k to provide an interface with other systems on capability i.
    - growth_rate: the continuous growth rate per time unit. It must be positive to ensure that the additional time needed is minimized.
    - pm: the required profit margin of capabilities. pm describes the minimum required rate of return from providing capabilities. It is a vector of n elements. "pm(i)=0" if "y(i)=0". pm is nonnegative. It is the primary parameter for modeling the utility function of the system k.
- If: pm(i) = 0, the system k would like to collaborate with SoS in providing capability i without an attempt to make profit (but losing profit is not acceptable).
- Otherwise: the system k would like to collaborate with SoS in providing capability i only if the minimum required rate of return is met (i.e. making a minimum level of profit is required from the collaboration). The greater the pm, the "greedier" the system k is.
- CriticalPro: the probability of sending the P1 result to SoS, and (1-CriticalPro) is the probability of sending the P2 result to SoS. CriticalPro is within the range of [0,1], including the two

boundaries. Use CriticalPro to control the way of negotiation: (P1) or (P2). There are two extreme scenarios:

- If: CriticalPro = 0, the result of (P2) is sent to SoS;
- If: CriticalPro = 1, the result of (P1) is sent to SoS;
- Notes:
- (P1) tells the best performance of the system k at the given deadlines and funding.
- (P2) tells the additional funding needed and additional time needed in order to meet the performance requirements (if the system k is capable).
- In (P2) the additional funding needed may be greater than the minimum additional funding needed; similarly, the additional time needed may be longer than the minimum additional time needed. This is a strategy that the system k uses in the negotiation and protecting its private information. Please refer to parameters AF1, AF2, and AT for the details.
- Negotiation Parameters for (P2)
    - AF1: If the system k will use up all funding to be provided by SoS, it requests up to AF1 more than the additional funding needed. AF1 is a nonnegative real number. The greater the AF1, the greater the profit, and the failure rate as well, from the negotiation with SoS.
    - AF2: if the system k will not use up all funding to be provided by SoS, it requests up to AF2 more than the funding provided by the SoS. AF2 is a nonnegative real number. The greater the AF2, the higher the failure rate and profit from the negotiation with SoS.
    - AT: the system k request up to AT units of time more than the minimum additional time needed. AT is a nonnegative integer.
- If: AT=0, only the minimum additional time needed is requested.
- else: request up to AT units of time more than the additional time needed.
- The greater the AT, the higher the failure rate and time flexibility from the negotiation with SoS.

3.5.2 Cooperative System Negotiation Decision Model

Figure 13 illustrates the bilateral negotiations for the cooperative decision model. The protocol used of the cooperative negotiation model falls within the class of multi-issue negotiations, where the agents negotiate on all the issues together. The strategy deployed here is to negotiate bilaterally between the participating system agent and the SoS agent. This incorporates estimating functions and weighted counter proposals. It is assumed the agents following cooperative negotiation protocol have an innate behavior of being helpful and supportive to the SoS formation. However, at the same time there are certain constraints that need to be incorporated in their overall response to the SoS offer. Thus, this protocol presents a semi-cooperative negotiation model, which models the tradeoff between the two behaviors for the agents. A weighted sum approach is used to arrive at the final proposal value. It is assumed that different systems will weigh the two behaviors differently.

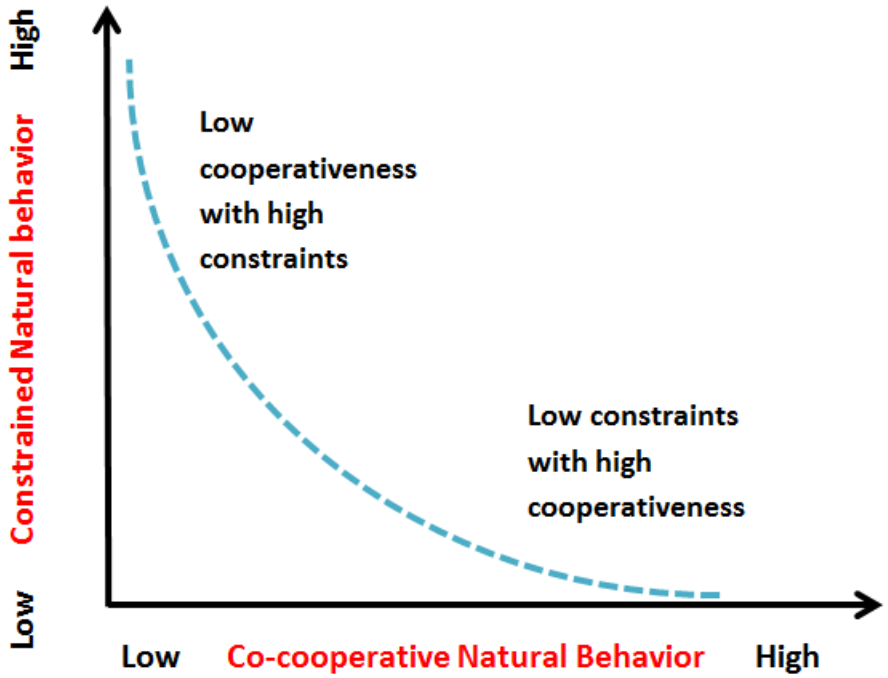

Figure 13 Compromise between innate behavior and constrained behavior

Since each system can be asked for providing one or more capabilities, represented as Ci subscript $i \in (1,2 \ldots N)$. The attributes to be negotiated in this model are performance, funding and deadline, denoted by $Aj, with$ the subscript $j \in (1,2,..M)$, respectively. Maximum and minimum values for issues under negotiation are also defined as (min_j, max_j) for issue j. $Aij\ is\ referred\ to\ as\ attribute\ j\ under\ negotiation\ for\ capability\ i$

$$Aij_p\ is\ attribute\ performance\ for\ capability\ i\ ,$$

$$Aij_d\ is\ attribute\ deadline\ for\ capability\ i,$$

$Aij_f\ is\ attribute\ funding\ for\ capability\ i$, where jp, jd,jf ϵ j.

The number of systems is S denoted by t.

$Iitt'\ is\ the\ interface\ between\ system\ t\ and\ t'\ required\ by\ SoS\ for\ capability\ i, where\ t, t'$ ϵ (1, 2..S) and t≠t'

Since the amount of resources is fixed for an individual system, it has to divide them in order of preference of capability production/ development.

It is assumed a system has 'R' units of resources to be divided among as Ci capabilities.

To get an order of preference, the following metric can be proposed:

$$\text{metric } Mi = \left(\frac{Aij_p}{Aij_f}\right) * \sum_{t',t\neq t'} Iitt'$$

A ratio of metrics is taken to calculate the amount of resources to be distributed amongst the capabilities for development. Metric Mi can be explained as amount of performance required per unit of funding and the complexity of the task.

$$Each\ capability\ i\ is\ given\ Ri\ amounts\ of\ resource\ such\ that\ Ri = \left(\frac{Mi}{\Sigma Mi}\right) * R \text{ and } \Sigma Ri = R$$

To calculate the resources needed for development of each capability, the above equation is proposed. This equation ensures that resources are distributed based on performance required for $capability\ i$, funding provided for $capability\ i$ and the complexity of task. Since the system cannot increase the amount of resources during negotiation in the current epoch, it is defined that the sum of resources for all capabilities is equal to the total resources at all times.

Complexity of the task or capability i is estimated by the variable Coi. Coi is defined as the ratio of the number of interfaces require by a system for a capability I development to the total number of interfaces possible.

The total number of interfaces is calculated as total systems participating subtracted by one. This reduces the estimation of the real complexity of task.

$$Total\ number\ of\ interfaces\ possible\ for\ each\ capability\ required = S - 1$$

$$number\ of\ interfaces\ required\ by\ SoS\ for\ capability\ i = \sum_{t=1,t\neq t'}^{S} Iitt'$$

$$Coi = \frac{number\ of\ interfaces\ required}{Total\ number\ of\ interfaces\ possible}$$

Number of interfaces in the current epoch is denoted by I_c

Number of interfaces in the previous epoch is denoted by I_p

Maximum number of interfaces possible is denoted by I_{Max}

Cooperative Estimation of Attributes for Negotiations:

An innate cooperativeness factor (V) and constrained innate factor (N) also need to be defined. The cooperativeness factor represents the level of cooperative behavior in a system. The constrained factor defines an evaluation of the importance of the issue to the system in the current epoch of negotiation.

For each issue a priority function is defined based on the current epoch, complexity and the number of changes in interfaces required by the SoS manager from the previous epoch. This becomes:

$V \in (0,1]$; V varies for each individual system depending on the cooperativeness. A value of $V = 0$ denotes complete cooperation and 1 denotes least cooperation.

The deadline estimated by the agent for capability i , through the cooperative factor isAij_{vd}, where $Aij_{vd} = [(\frac{\boldsymbol{V}}{\boldsymbol{Ri}}) * Aij_d]$ (next higher integer value)

The funding estimated by the agent for capability i , through the cooperative factor is Aij_{vf}, where $Aij_{vf} = (\frac{\boldsymbol{V}}{\boldsymbol{Ri}}) * Aij_f$

The performance estimated by the agent for capability i, through the cooperative factor is Aij_{vp}, where $Aij_{vp} = (\frac{\boldsymbol{V}}{\boldsymbol{Ri}}) * Aij_p$

These equations calculates the estimated deadline for $capability\ i$ based on cooperativeness of the system, deadline offered by SoS and the amount of resources allocated for$capability\ i$. A higher cooperation, which means the value of V is close to 1, can be translated for each attribute by multiplying it with the given offer. Similarly, the resources allocated will affect the estimation of attributes and hence are placed in the denominator of the metric. A higher resource allocation will thus tend to give a lower estimation of the attribute by the individual system and vice versa.

Constrained Estimation of Attributes for Negotiations:

Aij_{ld} represents the constrained function for negotiationg deadline

Aij_{lf} represents the constrained function for negotiationg funding

Aij_{lp} represents the constrained function for negotiationg performance

$$Aij_{ld} = \text{next highest integer value of} \quad [Aij_d{}^{e^{\frac{\boldsymbol{I_C - I_P}}{\boldsymbol{I_{max}}}}}]$$

The deadline is estimated here as a function of the amount of change required in interfacing with other system from the previous epoch. In this way the system can estimate how much more time it will need to fulfill the requirement of the desired capability. If the numbers of interfaces required are more in the current epoch, than the previous epoch the deadline increases. To normalize the effect, the total number or maximum number of interfaces possible divides the change. The exponential rose to this quantity and is then modeled as an exponent of the deadline required by the SoS. The value then calculated is rounded off to the next highest integer. The similarity of the interfaces in two consecutive epochs could be taken into account to measure the effort required by the system. The number of new interfaces required which are not similar to the one's in the previous epoch can also be taken into account for further improvement of the metric.

$$Aij_{lf} = Aij_f{}^{e^{\alpha(\#\ of\ epochs-1)}} \quad \text{where } \alpha = 0.1$$

For estimating the funding, the time value of money is considered as a primary concern for the system. As the number of epochs increase or number of negotiations increase, more time is taken to decide to the participation. This adversely affects the systems funding requirements. Hence funding offered is raised to the power of number of epochs-1 times a positive quantity taken as "alpha". The maximum value reached by the system in terms of funding can be the provided funding itself since if the number of epochs is 1, the exponential becomes 1 as well.

$$Aij_{lp} = \mathrm{Aij_p} * (\frac{\mathbf{1}}{\mathbf{1 - Coi}})^{\mathbf{1}}$$

The performance is estimated to decrease with increase in complexity. Hence, the metric proposed above is used to reflect that as the complexity increases, the performance goes lower than the required value. The maximum value reached can be the required performance itself.

The idea is to combine the values from both perspectives to give the agents a realistic picture of their estimates. Since this represents the cooperative model, it is assumed that cooperative behavior is more important than the constrained behavior. Therefore, to combine the values of negation issues from both of these behaviors, it is proposed to use a weighted aggregation approach.

This proposed aggregation uses a weights approach comprised of two inputs: the cooperative behavior value and the constrained behavior value. Normalized weights are defined as non-negative real numbers,

$w1, w2, \ldots, wm,\ such\ that \sum wi = 1.$

$$Proposed\ deadline = weighted\ average\ (cooperative\ deadline, constrained\ deadline)$$

$$\mathrm{Proposed\ deadline} = \mathrm{WA}(\mathrm{Aij_{ld}}, Aij_{vd}\ )$$

$$\Delta \mathrm{d} = \mathrm{deadline\ by\ SoS} - \mathrm{proposed\ Deadline}$$

Similarly, all other delta values are computed for funding and performance.

Basic assumptions for the individual systems decision making:

The individual systems will make decisions based on the following considerations:

1. The establishment of each demanded capability is an independent task for the individual system.
2. Individual system will concede based on the reply of SoS to the offer made. If the SoS concedes first it, the system accepts the last offer it made, else the system concedes after a certain number of negotiation cycles or epochs on its offer.

3.5.3 Opportunistic - Markov Chain Model

The model presented here for the system in question is an "opportunistic" model, i.e., that the system can behave selfishly as well as unselfishly (or selflessly). In other words, by tweaking a certain tunable parameter, η, an entire spectrum of behavior – ranging from extremely selfless to extremely opportunistic (selfish) – can be obtained from the system. Hence, in a sense the system's behavior can be characterized as "opportunistic" because there is no fixed pattern of behavior that this system will exhibit. It needs to be understood that how the system behaves can be controlled by varying the tunable parameter (η), and thus it is possible to have a large number of systems, using differing values of η, ranging from very selfish to extremely selfless. An example is provided at the end to illustrate the interaction process between the SoS and the system concerned.

The system will interact with the SoS as follows. When the SoS provides via a data structure the following information: an architecture, a desired performance level, the level of funding, and the deadline, the system will perform internal calculations to develop outputs for the following:

- Willingness of the system to cooperate with the SoS
- The performance level that the system can deliver
- The funding it will need
- The deadline by which it will be able to complete its task

A project management model based on Markov chains is used for estimating the above-mentioned outputs. The mathematical details of this model are now below. The opportunistic behavior of the system will be dependent on an adjustable *opportunistic parameter η*. The SoS will have the ability to change this parameter in order to develop a class of systems with differing behavior. Further, the sluggishness of the system will be defined by additional parameters (*l* and *m*), which also the SoS will have the ability of adjusting.

Methodology:

Essentially, the work assigned by the SoS to the system will be assumed to be a ``project'' that takes a random amount of time and a random amount of resources (funding) to complete. The internal phases of this project is modeled as a Markov chain, and the Markov chain will be to estimate the expected (mean/average) amount of time and funds needed by the system. The willingness to cooperate is based on how quickly the system will be able to complete the task (project) at hand and its own workload. In what follows, we provide the details of how these calculations will be performed to generate outputs to be supplied to the SoS.

The three internal phases in the project are: "initial," "intermediate," and "completion." Each of these will be considered to be states in an absorbing Markov chain in which the absorbing state is completion. We will assume that after unit time (e.g., one cycle), the project will move from one state to the next with a given probability. The input chromosome will be converted into a matrix of 0s and 1s, which will essentially represent the nature of interaction and the workload imposed on the system under consideration. These inputs will be used to compute the number of systems with which the current

system must interact. This number will be used to compute the probabilities of the Markov chain. From the one-step transition probabilities of the Markov chain, it is possible to estimate the mean amount of time needed for completion of the project. If the resources are assigned according to the worst-case scenario, the mean time of the project completion will be used to estimate the level of cooperation.

Notation:

- n: number of systems in SoS
- i: index of the system under consideration, where $i = 1,2, \ldots n$
- p_i: performance desired from system i by SoS
- C: system architecture (interface): chromosome (matrix of 0s and 1s)
- d_i: delivery deadline set for system i by SoS
- f_i: funds allocated to system i by SoS; values for $i = 1,2, \ldots n$ will be provided to the ith system
- Δp_i: the difference in the performance that system i expects to deliver
- Δf_i: the difference in the funding that system i requires
- Δd_i: the difference in the deadline by which system i expects to deliver
- η: selfishness parameter in the interval (0,1] that SoS can adjust to obtain a spectrum of participating systems ranging from very unselfish to very selfish; 0 being very selfish and 1 being very unselfish (selfless)
- l and m: additional system behavior parameters, each taking values in the interval (1,2], which the SoS can change to obtain a whole spectrum of participating systems ranging from very fast to very sluggish; 1 being very fast and 2 being very sluggish.

We will need some additional notation required in the internal calculations of the model that we will define later. The Markov chain of the internal system dynamics will be defined by the following transition probability matrix:

$$\begin{bmatrix} \frac{k}{ln} & 1-\frac{k}{ln} & 0 \\ 0 & \frac{k}{mn} & 1-\frac{k}{mn} \\ 0 & 0 & 1 \end{bmatrix}$$

where state 1 is the initial phase, state 2 the intermediate phase, and state 3 the completion phases. In the above, k will denote the number of systems with which system i will interact. The parameters l and m will be additional system behavior parameters that the SoS will be able to change at will. The value of k is computed from the chromosome C. The, using standard analysis of an absorbing Markov chain [141], the number of expected cycles needed to complete the project starting at the initial phase will be computed. This number will be denoted as τ_i for the ith system. We now explain how the absorbing Markov chain analysis is performed. We construct the so-called Q-matrix from the transition probability matrix as follows eliminating the completion phase:

$$Q = \begin{bmatrix} \frac{k}{ln} & 1 - \frac{k}{ln} \\ 0 & \frac{k}{mn} \end{bmatrix}$$

Then, the following operation is performed:

$$M = I - Q$$

where *I* denotes the identity matrix. Let *N* denote the inverse of *M*. Then, the expected numbers of cycles to completion are computed as follows:

$$T = N \cdot C$$

where T and C are column vectors. C is a column vector of ones, and T(1), the first element in the column vector, will equal τ_i. Setting *k = n* and repeating the calculations above, we will obtain an upper limit on the expected number of cycles needed for completion; this limit will be denoted by τ_{max}. The willingness to cooperate will then be computed as:

$$\Phi = 1 - \frac{\tau_i}{\tau_{max}}$$

Then, the actual performance of the system will be given as

$$p_i \, \Phi$$

which would imply that the change in performance will be given by:

$$\Delta p_i = p_i(\Phi - 1)$$

Further, the deadline, i.e., the number of cycles, by which the system will be able to deliver, will be computed as

$$\frac{d_i}{\Phi}$$

The above scalar is rounded up to the next higher integer. The above implies that the requested change in the deadline will be:

$$\Delta d_i = d_i(\frac{1}{\Phi} - 1)$$

And finally, we now show how the requested change in the budget will be computed. We use $\bar{B}$ to denote the upper limit on the budget, which will effectively serve as its own estimate of the upper limit, which is computed as follows: will be computed as:

$$\bar{B} = \frac{f_i}{\eta}$$

where η is the *opportunistic* parameter. For a very selfless system (η =1), $\bar{B}$ will be computed as the actual funds provided divided by this opportunistic parameter. A very selfless system will assume that the funds provided to itself form the upper limit and that there are no other funds available for use. A selfish system with η <1 will assume that the limit is (1/η) times the funds provided to itself. Then, the maximum rate at which the SoS can fund projects will be computed as:

$$\rho = \frac{\bar{B}}{\tau_{max}}$$

Then, the actual funding needed by the system will be computed as

$$\tau_i \rho$$

This would imply that the change requested in the funding will be:

$$\Delta f_i = \tau_i \rho - f_i = \tau_i \frac{\bar{B}}{\tau_{max}} - f_i$$

Setting η to a very small positive number close to 0 produces a system that is very selfless, while setting it to 1 produces a very selfish system. Thus, a large number of systems can be generated during the negotiation process by assigning a range of values to η, e.g., 0.01, 0.02, 0.03, 0.1, 0.3, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99, 0.999. Also, by varying the values of *l* and *m* in the range (1,2], one can obtain a range of systems whose ability to meet deadlines and deliver performance levels varies from very fast (e.g., 1.01) to very sluggish (2

4 Implementation of Multi- Agent Based Architecture Model for Acknowledged SoS

4.1 Agent Based Model Framework (ISR)

This phase of the report describes the implementation of the wave model in ABM for the ISR framework. Two domains: ISR and SAR were studied in the architecture selection part of the program. ISR is implemented through the ABM portion. Figure 14 describes the transition states of the SoS agent and the system agents. The transition states of the SoS agent are analogous to Dahman's wave model

proposed for Acknowledged SoS. The wave planning methodology was originally proposed by Dombkins [155].

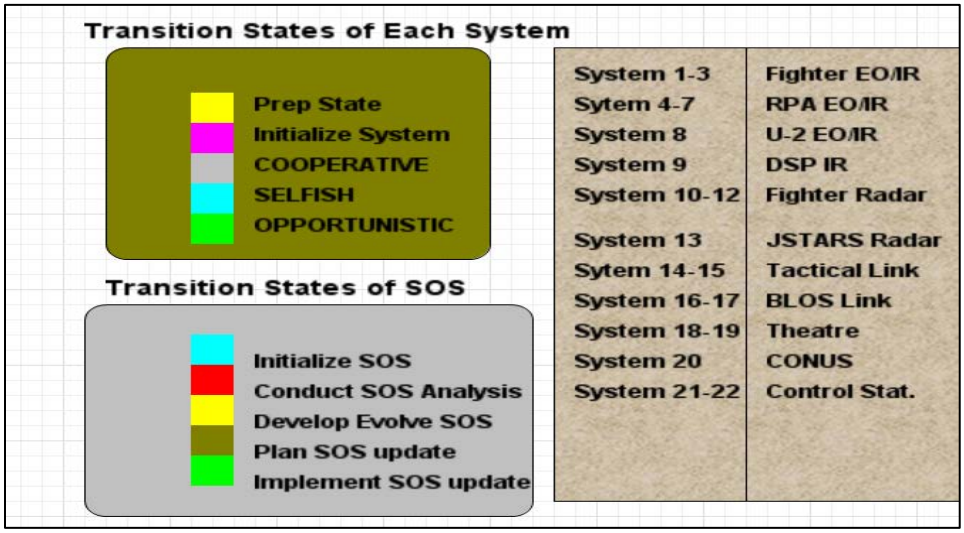

Figure 14. Transition states of the ABM, and member system types

Figure 14 is the part of the user interface of the ABM. On the left can be seen different transition states of each systems agent and SoS agent. Each state is represented by a different color to make it easier to recognize the state of each agent while the model is running. On the right in the figure, the names of each system are printed along with the corresponding system number.

The table 5 reflects how many capabilities each participating system possesses. For example, System 1 through 7 possess capabilities C1 and C5, System 8 and 9 hold just one capability C1. System 10 through 13 holds capabilities C2 and C5 and so on.

Table 5. Display of systems with their corresponding capabilities

System Numbers	Capabilities possessed
System 1- System 7	(1 and 5)
System 8- System 9	(1) only
System 10- System 13	(2 and 5)
System 14- System 16	(3 and 5)
System 17- System 18	(4 and 5)
System 19- System 22	(5) only

The ABM incorporates three techniques of meta-architecture generation namely fuzzy-genetic approach, multi-level optimization and multi-objective optimization. The user (Acknowledged SoS manager) interface initially allows choosing the preferred meta-architecture generation technique for the domain specific information. The user can then define the preferred architecture quality on a scale of 1 to 4 in the continuous domain as a threshold for the final architecture. Number of negotiation cycles that the SoS agent can have with the individual system agents can also be predefined by the user. Other parameters, which are user-defined, include:

1. Setting the parameter values for the 3 individual system negotiation models
2. Parameters for SoS agent negotiation model

Figure 15 and 16 show the user interface for entering input values required to run the ABM framework.

The SoS agent state chart can be visualized in Figure 16. The directed arrows indicate the transition direction of the agent from one state to the other. After entering the required inputs to run the model, ABM initializes the SoS agent. This can be visualized as the SoS agent, which is represented by a circle on the interface, turns cyan in color. Consequently, it quickly transitions to Conduct_SoS_Analysis state. During this state, the SoS agent executes the meta architecture generation procedure selected by the Acknowledged SoS manager. This also starts the first wave of the ABM model. The execution outputs a meta-architecture in the form of a genetic chromosome as described in section 3.2.3. Using the meta-architecture generated the procedure creates 22 system files. These files contain the information that needs to be disseminated to the system agents. Figure 17 illustrates the information given to system 1 for acquiring capabilities. Since system 1 possesses 2 capabilities which are C1 and C5, hence only 2 rows contain information on deadline, performance, and funding.

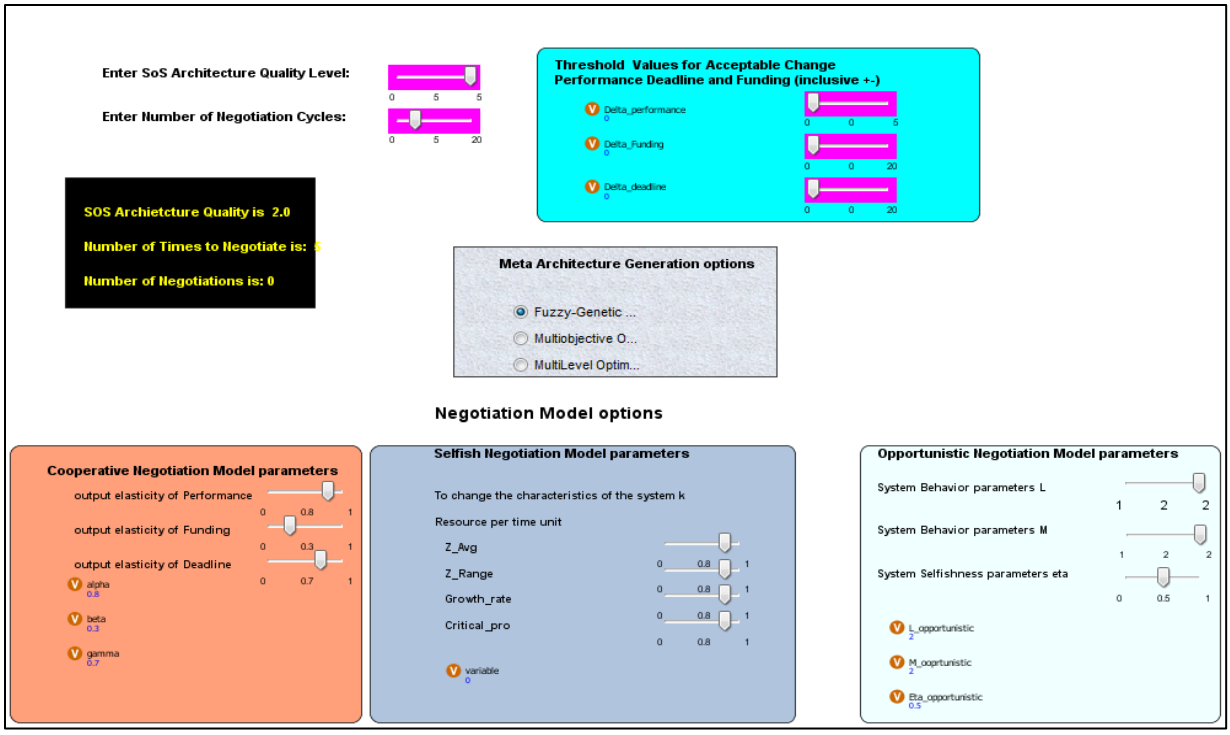

Figure 15. ABM User Interface

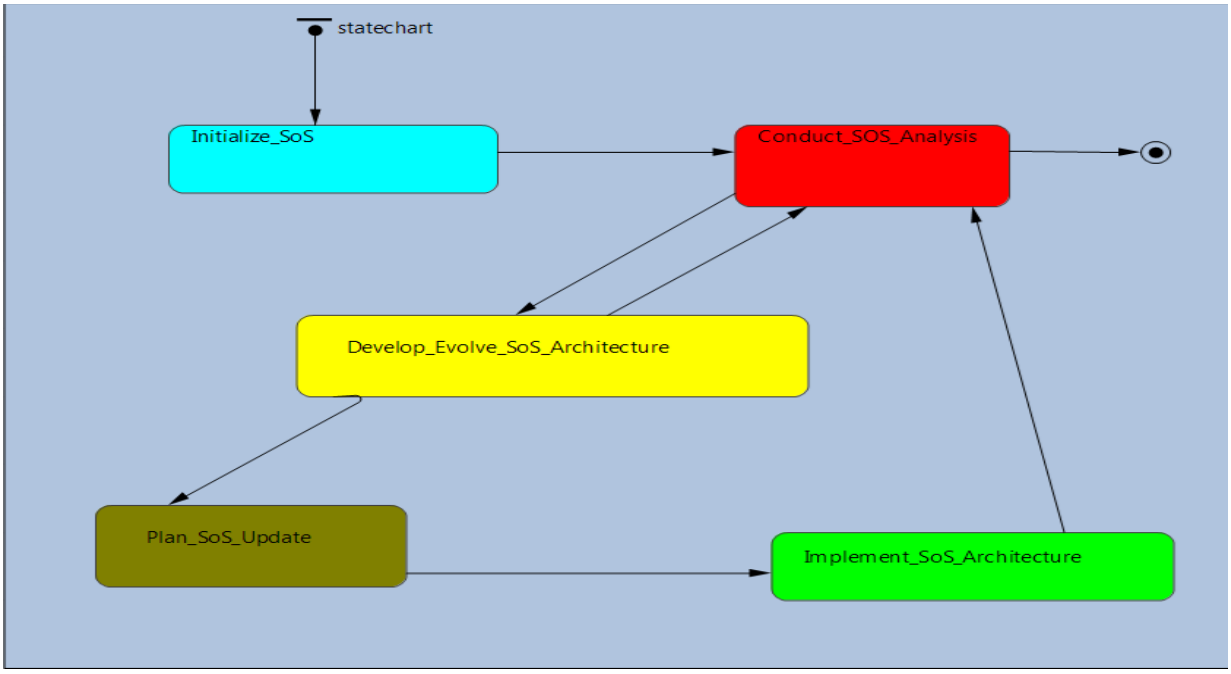

Figure 16. ABM SoS Agent Statechart

Once the SoS agent has completely executed the meta architecture generation process it transitions to Develop_Evolve_SoS_Architecture state. The SoS agent sends a message to all individual systems to transition them from the Prep state to Initialize_System state. During this stage the system agents get activated and SoS agent disseminates the information in the form shown below to acquire capabilities.

This also starts the first epoch of negotiation between SOS and System agents.

	Architecture (the portion related to system j)																							deadline	funding	performance
	S_j	$I_{j,1}$	$I_{j,2}$	$I_{j,3}$	$I_{j,4}$	$I_{j,5}$	$I_{j,6}$	$I_{j,7}$	$I_{j,8}$	$I_{j,9}$	$I_{j,10}$	$I_{j,11}$	$I_{j,12}$	$I_{j,13}$	$I_{j,14}$	$I_{j,15}$	$I_{j,16}$	$I_{j,17}$	$I_{j,18}$	$I_{j,19}$	$I_{j,20}$	$I_{j,21}$	$I_{j,22}$	$SoS.d_j$	$SoS.f_j$	$SoS.p_j$
C_1	1	0	1	1	1	0	0	0	0	0	1	0	1	0	1	0	0	0	0	1	0	0	0	1	10.1	21
C_2	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0			
C_3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0			
C_4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
C_5	1	0	1	1	1	0	0	0	0	0	1	0	1	0	1	0	0	0	0	1	0	0	0	1	10.2	23

Figure 17. Input file format for capabilities, a system and its interfaces, and deadline, funding and performance expectation

Figure 18 illustrates the state chart of individual system agents. The individual systems agents are preselected to have either of three negotiation models. On receiving the information, they execute their negotiation models to estimate requirements in terms of funding, performance and deadline for the development of required capabilities. The state of System agents can be visible on the ABM interface by the color of the small oval shape (represents the individual system agents). The systems reply with an offer in terms of difference of provided attribute values and the required amount. These are termed as delta funding, delta deadline, and delta performance respectively. A graphical presentation can be observed to gauge the comparison in delta values and actual values provided by the SoS as illustrated in Figure 19.

After SoS receives the reply from all individual system agents, the first epoch ends. Now SoS executes its negotiation model to decide which system should be a part of the overall architecture based on the delta values. SoS agent forms a new architecture based on its negotiation and evaluates its quality through the fuzzy assessor. If the architecture quality is below the threshold set by the Acknowledged SoS manager, SoS agent goes for another round of negotiations. This starts the second epoch. As explained above each negotiation initiates similarly. At the end of each negotiation offer and reply, the SoS agent decides to select system for participation in the overall mission and calculates the architecture quality. The negotiation keeps on going until maximum number of negotiations has been reached or the architecture quality is greater or equal to the threshold value. This also concludes the first wave of the ABM.

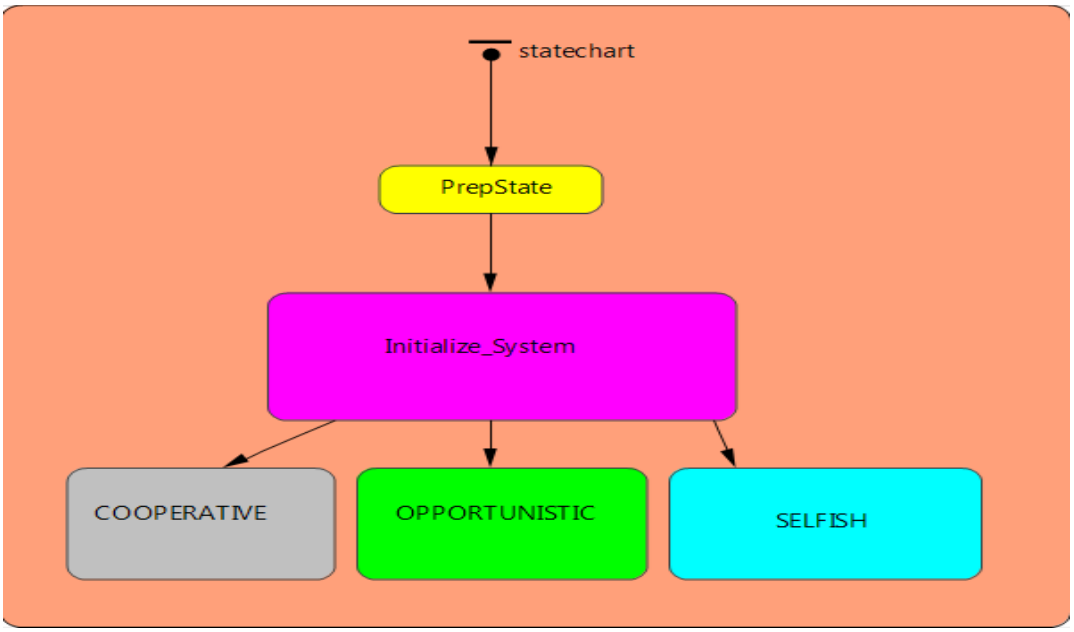

Figure 18. ABM Individual System Agent sate chart (Negotiation Model Choices for Systems)

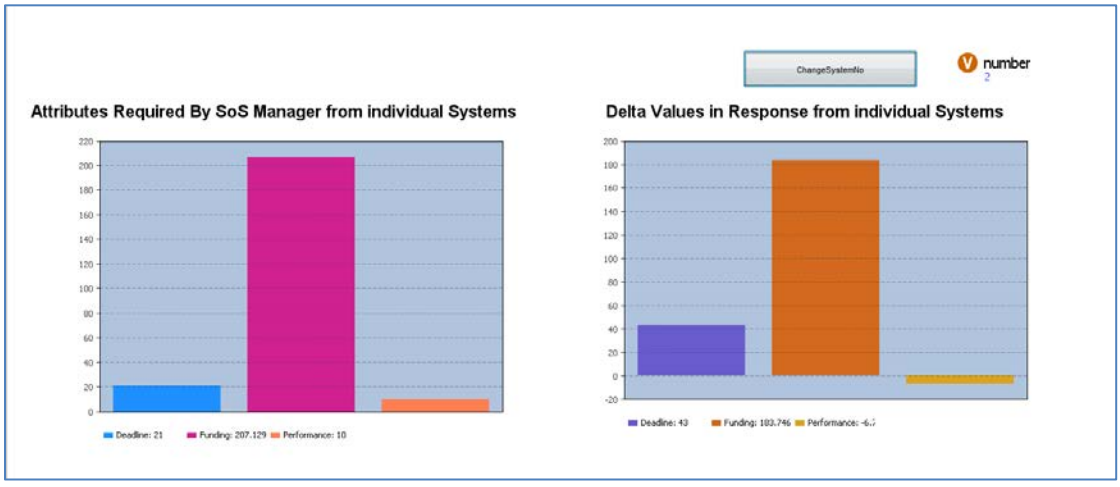

Figure 19. Attributes Status before and after negotiations

In case the SoS architecture quality is not able to reach the threshold value after the maximum number of negotiations, SoS agent transitions back to Conduct_SoS_Analysis state. SoS then re-executes a meta-

architecture generation procedure to come up with a new meta-architecture. This helps in re-exploring the trade space for another possible solution.

After he final architecture for the current wave has been selected the SoS agent transitions to the Plan_SoS_Update state. During this stage SoS plans for the next wave cycle, upgrades and reevaluates its parameters. Finally, SoS transitions to the Implement_SOS_Architecture state. The architecture is ready to be implemented by the Acknowledges domain manager. The next wave starts from the point where the first wave ends. The final architecture obtained in the previous wave becomes the commencing point for the next wave of modeling.

There were two challenges to the Agent-Based Model: Generality and Integration. Generality meant creating and ABM that would be applicable and adaptable to any domain. Integration meant interfacing the ABM with the three negotiation models namely selfish, opportunistic and cooperative, developed in MATLAB.

4.1.1 Generic Agent-Based Model

As mentioned in [9], the SoS development follows a Wave Model. The ABM was built based on this Wave Model and the ABM reflects the behaviors inherent in an Acknowledged SoS [8]. This research showed that the domain specific area in an Acknowledged SoS development was in the evaluation of an SoS architecture. This SoS architecture evaluation is done in the Fuzzy Assessor and the domain specific part of the Fuzzy Assessor are the Fuzzy Rules. In order to make the ABM generic to any domain, the Fuzzy Assessor was implemented in the SoS agent using a Fuzzy Associative Memory (FAM). The SoS agent reads in the FAM from an Excel file during initialization. In this way, a user can specify the domain specific Fuzzy Rules in an Excel file and the ABM can simulate the SoS development for that domain.

4.1.2 ABM Integration

Multiple models were developed as Matlab executable's independently by different researchers. The second challenge was how to integrate these models with the ABM. The models developed in Matlab were:

- SOS Meta-Architecture Models (3)
- Fuzzy Assessor
- SoS Negotiation Model
- Selfish System Negotiation Model
- Opportunistic System Negotiation Model
- Cooperative System Negotiation Model

The ANYLOGIC software used for agent based modeling has the capability to interface with excel files. Hence, the interface between the above mentioned models and ABM was through Excel files. All Models were called as an executable through Anylogic.

At the beginning of the epoch, the ABM calls the Meta-architecture generation executable file to get the initial SoS architecture and is recorded in a Excel file. This executable writes the initial performance,

funding, and deadline for each system to an Excel file. In order to prevent possible file access contentions, there is one Excel file for each system in the simulation that contains the performance, funding, and deadline for that system. The fuzzy assessor executable computes the architecture quality and values for each of the key performance attribute of the SoS. The overall SoS fitness measure is used within both the GA and the ABM portions of the framework to compare different chromosomes, or examine the impact of changes to negotiation rule sets or environment values for the same chromosome, and other products of analysis.

After the SoS reads the initial SoS architecture from the SoS architecture Excel file, the SoS then sends the Request for Connectivity message to each system in the simulation. There is one instantiation of the system agent for each system in the simulation. Funding is proposed by the SoS Agent once per epoch to each System. The SoS agent may include information about the desired interfaces in the allocation to each System and Capability, but funding is not spread directly to the interfaces by the SoS Agent, only to the System. Deadlines and Performance are handled the same way, one for each System and Capability.

Once the system has received the Request for Connectivity message from the SoS, the system calls the Matlab executable file that contains the system negotiation model assigned to that system. The results are placed in a second sheet named as output on the same excel file. The SoS agent read the excel file to run the SoS negotiation executable. The final architecture is placed in the excel file and is represented as a network of nodes and arc as shown below in Figure 20 which also shows the integration.

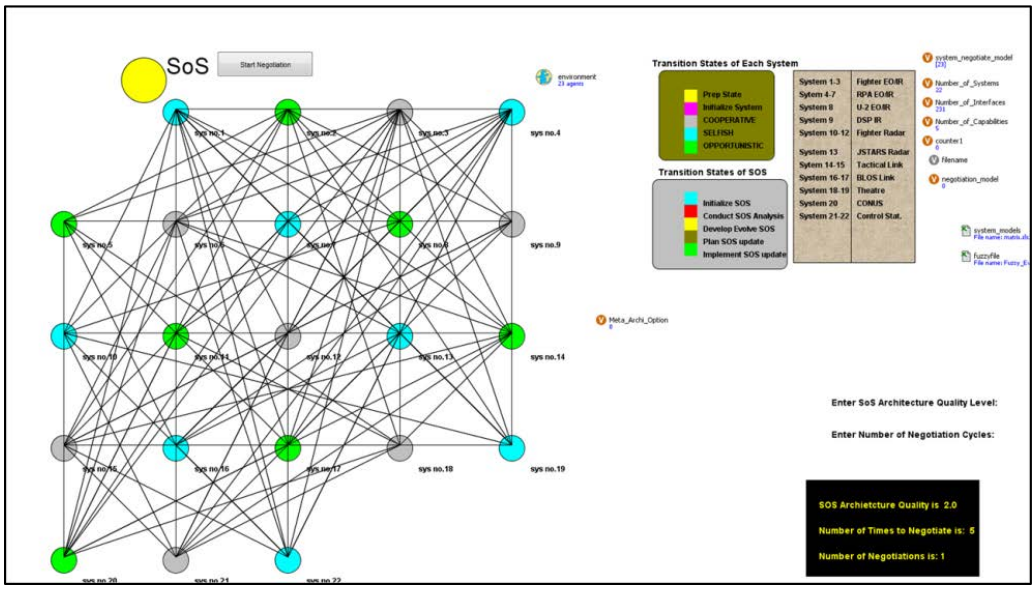

Figure 20. Overall Agent-Based Implemented Model Architecture

4.2 ISR and SAR Demonstration Domain Information

4.2.1 Historical Example

A guiding physical example is taken from history. During the 1991 Gulf War, Iraqi forces used mobile SCUD missile launchers called Transporter Erector Launchers (TELS) to strike at Israel and Coalition forces with ballistic missiles. Approximately 50-60 TELs were hidden in the western Iraqi desert, from which Iraqi forces launched over 200 missiles during the 30 day conflict. The Iraqi forces had developed new techniques called "shoot and scoot" that allowed them to reduce the TEL vulnerability time to half an hour to come out of hiding, set up, launch, and return to their hiding places. This was only one third of pre-war estimates, and a great surprise to Coalition planners [156]. While the relatively inaccurate Scuds were not a tactically significant factor in the war, they had a significant strategic impact on morale and cohesiveness of the Coalition. Therefore, the TELs became a "high value, fleeting" target.

Existing intelligence, surveillance, and reconnaissance (ISR) assets were inadequate to find the TELs during their vulnerable setup and knock down time. The "uninhabited and flat" terrain of the western desert was in fact neither of those things, with numerous Bedouin goat herders and their families, significant traffic, and thousands of wadis with culverts and bridges to conceal the TELs and obscure their movement. In addition, Iraqi forces produced some very fine camouflage and realistic decoys [157]. Even though thousands of sorties were flown, during hundreds of firing opportunities, TELs were spotted only 11 times, and the contacts were lost before completing an attack eight of those 11 times. The average time between spotting and arriving at a potential target was 90 minutes, which might have been marginally acceptable before development of the shoot and scoot tactic [156]. This offers a prime example of existing systems being inadequate to address a mission, but some relatively low cost, quick changes, and joining together of existing Systems might have been used to create an SoS capability to achieve the mission.

Applying the method described in section 2.5.3 above to the ISR problem resulted in the following input domain parameters.

4.2.2 ISR Domain model

The characteristics of the SoS reached by consensus of stakeholders and subject matter experts (SMEs) are listed in Table 6.

Most important requirements of the ISR SoS reduced down through the SME discussions to the following four attributes, measurable by operations on the chromosome describing the SoS:

- **Performance** as the sum of the square miles of terrain searched and targets found per hour, divided by the search area
- **Affordability** given by total cost ranges of development and operation of the SoS; less cost is more affordable

Table 6. ISR SoS domain example characteristics

Overarching Purpose of SoS	ISR & Targeting of Gulf War Scud TELs	
Unique value of SoS	Existing non-networked systems not doing job	
SoS Measures of Effectiveness	Probability of successful engagement per day	
Issues that might limit effectiveness	SCUD TEL concealment and countermeasures Short time of exposure of TEL before and after launch	
SoS features that might greatly increase effectiveness	Improved probability of detection in presence of concealment Significantly Improved speed of response	
Desired Effectiveness	About 1 successful engagement per day or more	
Stakeholders	Operating commands, system operators/crew/maintainers, intel agencies, coalition partners, regional states, system program offices, troops in theater, contractors, Congress, DoD, enemy forces	
ROM Budget: Development	About $40 Million	
ROM Budget: Operations	About $40 Million	
Attributes of the SoS, and range limits for fuzzy evaluation	Performance Affordability Robustness Flexibility	Find and attack within half a hour; find >1 per day <$100M for development and thirty day operation <15% performance loss for loss of any single system Very few single source per capability
Capabilities of contributing systems	EO/IR Synthetic Aperture Radar Communications Command & Control Exploitation	

- **Flexibility** in terms of development –multiple sources are available for each capability
- **Robustness**, defined as the smallest maximum loss of performance by successive removal of each participating system [158] [37]

Performance and affordability are augmented by a factor depending on the interconnectedness (interfaces) represented by the chromosome.

The capabilities of the ISR SoS, contributed by the constituent systems were broken down as:

- Electro-Optic/InfraRed (EO/IR) search capability

- Side looking, synthetic aperture radar (SAR)
- Command and control facilities
- Exploitation centers (smaller ones in theater and a large one in CONUS)
- Communication capabilities, both line of sight (LOS) limited to in-theater, and beyond line of sight (BLOS)

Taking a slight poetic license with respect to the historical example, we proposed the following types of systems, with the non-communication systems limited to one primary capability plus communications: (i) **Fighters**, some equipped with an EO/IR capability, some with SAR, (ii) Remotely Piloted Aircraft **(RPA)**, equipped with better EO/IR capability, (iii) **U-2** aircraft, primarily equipped with EO/IR capabilities, but limited to film, so that system is not timely, but can help reduce the overall search area for the other systems, if it participates, (iv) Defense Support System **(DSP)**, that can survey the entire area, but only provide notice on actual launch, reducing the time for the fighters to arrive before they're hidden again, (v) **JSTARS**, with large SAR, (vi) **Control Stations** for the RPA, (vii) ISR data **Exploitation** centers, (viii) Communication systems, LOS and BLOS, that enable the interaction between systems that make the SoS work.

Working from the ground up, a possible set of capabilities and costs of systems and interfaces for a SoS to address the Gulf War TEL problem are shown in Table 7. This resulted in an ISR SoS with 22 potential systems of 9 types, with 5 different capabilities among them, with at most 2 capabilities per system.

Table 7. Domain model of SoS with 22 Systems: Capabilities, Costs, and Schedules

System	Type Sub-System	Capability Number	Coverage sq mi/hr;	Develop $M/ epoch/ interface	Operate $K/hr per system	Time to Develop, Epochs	Number possible in SoS	System Number
Fighter	EO/IR	1	500	0.2	10	1	3	1-3
RPA	EO/IR	1	2000	2	2	1	4	4-7
U-2	EO/IR	1	50000	0	15	0	1	8
DSP	IR	1	100000*.01	1	1	1	1	9
Fighter	Radar	2	3000	0.7	10	1	3	10-12
JSTARS	Radar	2	10000	0.1	18	1	1	13
Theatre	Exploit	4	5000	2	10	1	2	14-15
CONUS	Exploit	4	25000	0.2	0	0	1	16
Control Station/ AOC	Cmd & Control	5	1	1	2	1	2	17-18
LOS Link	Comm	3	1	0.2	0	1	2	19-20
BLOS Link	Comm	3	1	0.5	3	1	2	21-22

The Inputs from Table 7 were adjusted slightly to simplify the model by scaling all the capability contributions to be relative to square miles searched per hour. This allowed a simpler performance

algorithm to be implemented in the fuzzy fitness assessor. The equivalent input data to Table 7 are shown in the Excel input sheet shown in Figure 21.

Name	ISR										
NumSys	22	m									
NumCap	5	n	*sys has capability, costs, perf, deadline*				*1*	*2*	*3*	*4*	*5*
SysNo	**Type**	**Capability**	**I/FDevCos**	**OpsCost/I**	**Perf**	**DevTime**	*EO/IR*	*SAR*	*Exploit*	*C2*	*Comm*
1	fighter	1	0.2	10	10	1	x				x
2	fighter	1	0.2	10	10	1	x				x
3	fighter	1	0.2	10	10	1	x				x
4	RPA	1	0.4	2	10	1	x				x
5	RPA	1	0.4	2	10	1	x				x
6	RPA	1	0.4	2	10	1	x				x
7	RPA	1	0.4	2	10	1	x				x
8	U2	1	0	15	3	0	x				
9	DSP	1	1	0.1	8	1	x				
10	ftrSAR	2	0.7	10	15	1		x			x
11	ftrSAR	2	0.7	10	15	1		x			x
12	ftrSAR	2	0.7	10	15	1		x			x
13	JSTARS	2	0.1	18	40	1		x			x
14	ThExp	3	2	10	10	1			x		x
15	ThExp	3	2	10	10	1			x		x
16	ConUS	3	0.2	0.1	15	0			x		x
17	CmdCont	4	1	2	12	1				x	x
18	CmdCont	4	1	2	12	1				x	x
19	LOS	5	0.2	0.1	10	1					x
20	LOS	5	0.2	0.1	10	1					x
21	BLOS	5	0.5	3	10	1					x
22	BLOS	5	0.5	3	10	1					x

Figure 21. ISR domain specific input data

The binary matrix of capabilities contributed by systems is shown in Figure 22 . It is equivalent to the x's in the cells on the right side of Figure 21. The ISR model of 22 systems is the one currently being implemented further in the ABM model.

Capability	CapName	*Cap-Sys1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*	*19*	*20*	*21*	*22*
1	EO/IR	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
2	SAR	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0
3	Exploit	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0
4	C2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
5	Comm	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 22. Binary matrix of capabilities vs. systems

4.2.3 Search and Rescue (SAR) Domain

We were interested in applying this method to a second domain to insure the fuzzy evaluation and GA worked as planned. A Coast Guard Search and Rescue (SAR) problem serving the Alaskan coast was selected. When there is a vessel in distress, the law of the sea requires other mariners to go to its aid.

The Coast Guard has numerous systems with differing capabilities such as cutters, aircraft, helicopters, communication systems, and control centers available from several stations in the area. In addition, fishing vessels, civilian craft, and commercial vessels join in an ad hoc SoS to provide assistance when a disaster strikes. The method described in section 2.5.3 was applied again. Background information was gathered (references) and SMEs were consulted. A sample SAR SoS with 29 systems of 9 types, with 10 different capabilities, with as many as 9 capabilities per system is shown in Figure 23 and 24.

The Search and Rescue (SAR) mission aims to minimize loss of life, injury, and property damage or loss at sea by finding and providing aid to those in distress. The SAR mission framework is inclusive of many activities from conducting search planning and coordinating SAR response, actual searching for, locating, and rescuing mariners and others in distress, providing necessary medical advice, assistance, or evacuation, and provide, when necessary, persons in distress safe transport to shore. Various components, such as Coast Guard cutters and helicopters, commercial and private sea vessels, Unmanned Vehicles (UVs), and private pilots and aircraft have some reconnaissance capability that may be brought together in a System of Systems (SoS) construct to assist in this ever evolving mission. [159] [160]

As defined in the National Search and Rescue Plan, ref (a), and Supplement, ref (b), participating search and rescue organizations may obtain permissible assets within the required SAR regions at any notice. These regions include all waters subject to U.S. jurisdiction and international waters in the Atlantic, Pacific, and Arctic Oceans and the Gulf of Mexico. Additional regions include identified Department of Defense (DoD) Area of Responsibilities (AORs). Partnerships exist among maritime industry in the Automated Mutual-Assistance Vessel Rescue (AMVER) system, and coordination among Federal, state, local, and tribal authorities to coordinate SAR operations is extensive. This section describes an example operational context for SAR missions, for which optimal SoS configurations can be determined given specific mission parameters and tradeoffs among SoS attributes such as performance, flexibility, robustness, and affordability. [159] [160]

Use of the Bering Sea and the Arctic by commercial fisheries, oil exploration and science is increasing. With the rise of the number of people and vessels in the area, the likelihood of a large SAR scenario occurring increases.

Possible missions related to this setting are summarized in Table 9:

Table 8. Characteristics of a SAR SoS

Overarching Purpose of SoS	1) Maritime Search & Rescue (SAR) of Bering Sea small airliner crash at sea 2) Stranded Cruise Ship in Other Territorial Waters 3) Find Two people in a small boat
Unique value of SoS	Greatly enhanced SAR Capability
SoS Measures of Effectiveness	Time to search 100,000 Sq Mi Probability of detection of survivors within 2 hours/within 12 hours
Issues that might limit effectiveness	Weather Availability of participant systems Language barriers Number of Survivors Sovereignty questions
SoS features that might greatly increase effectiveness	Speed of discovery Improved coordination of resources Ability to Prioritize resources at time of event, or during development
Desired Effectiveness	Find someone very fast and/or help lots of people relatively fast
Stakeholders	Federal, State, Local, Tribal, NGOs, Foreign, Crews, Survivor, Military, Coast Guard
ROM Budget: Development	$15M
ROM Budget: Operations	$10M
Attributes of the SoS, and range limits for fuzzy evaluation	Performance time to find someone before death by exposure or injury Affordability budgetary pressures, small civilian investment Robustness still works with only partial complement of systems Flexibility many choices of partners achieve all capabilities
Capabilities of contributing systems	EO/IR Night Vision Maritime Radar Emergency Locator Beacon System RF direction finder Deliver Paramedic/medical aid Remove survivor(s) to Emergency Medical Care Provide major medical capability Speed – Fast/Slow Time on Station Communications Command and Control/Coordination

Name	SAR							A								
NumSys	29															
NumCap	10						1	2	3	4	5	6	7	8	9	10
SysNo	**Type**	**Capability**	**I/FDevCos**	**OpsCost/I**	**Perf**	**DevTime**	*IR – range*	*Night Visi*	*Visual – ra*	*Maritime*	*RF Directic*	*Deliver Me*	*Remove s*	*Speed 300*	*Speed 15 r*	*Communic*
1	Cutter	7	0.03	2	12	1		x	x	x	x	x	x		x	x
2	Cutter	7	0.03	2	12	1		x	x	x	x	x	x		x	x
3	Helicopte	6	0.1	2	20	1	x	x	x	x	x	x	x	x		x
4	Helicopte	6	0.1	2	20	1	x	x	x	x	x	x	x	x		x
5	Aircraft	8	0.1	5	10	1			x		x			x		x
6	Aircraft	8	0.1	5	10	1			x		x			x		x
7	UAV	1	0.1	0.1	7	1	x		x	x			x		x	x
8	UAV	1	0.1	0.1	7	1	x		x	x			x		x	x
9	UAV	1	0.1	0.1	7	1	x		x	x			x		x	x
10	UAV	1	0.1	0.1	7	1	x		x	x			x		x	x
11	UAV	1	0.1	0.1	7	1	x		x	x			x		x	x
12	UAV	1	0.1	0.1	7	1	x		x	x			x		x	x
13	UAV	1	0.1	0.1	7	1	x		x	x			x		x	x
14	UAV	3	0.1	0.1	7	1	x		x	x			x		x	x
15	UAV	3	0.1	0.1	7	1	x		x	x			x		x	x
16	UAV	3	0.1	0.1	7	1	x		x	x			x		x	x
17	UAV	3	0.1	0.1	7	1	x		x	x			x		x	x
18	Fish Vesse	3	0.03	0.5	4	1			x	x		x	x		x	x
19	Fish Vesse	3	0.03	0.5	4	1			x	x		x	x		x	x
20	Fish Vesse	3	0.03	0.5	4	1			x	x		x	x		x	x
21	Fish Vesse	3	0.03	0.5	4	1			x	x		x	x		x	x
22	Fish Vesse	3	0.03	0.5	4	1			x	x		x	x		x	x
23	Civ Ship	7	0.05	2	8	1			x	x		x	x		x	x
24	Coord Ctr	5	0.05	0.5	5	1					x	x			x	x
25	Coord Ctr	5	0.05	0.5	5	1					x	x			x	x
26	Communi	10	0.02	0.03	1	0										x
27	Communi	10	0.02	0.03	1	0										x
28	Communi	10	0.02	0.03	1	0										x
29	Communi	10	0.02	0.03	1	0										x

Figure 23. The fuzzy assessor model inputs for the SAR SoS

Table 9. Possible SAR Scenarios

Scenarios	
1	A large sinking ship, cruise lines, or commercial freighter. Rescue of passengers, and/or a potential exposure of hazardous material (oil).
2	A ship stuck in the ice in the arctic ocean.
3	A commercial plane crash.
4	An oil rig disaster (explosion, etc.).

The basic conceptual radius of operation for the purposes of this application will include the Bearing Sea and the Gulf of Alaska as represented in the Figure 24. Extended loiter radii for airborne ISR mission profiles may extend the conceptual SAR mission profile to include the North Pacific Ocean, Chukchi Sea, Beaufort Sea, and Artic Ocean.

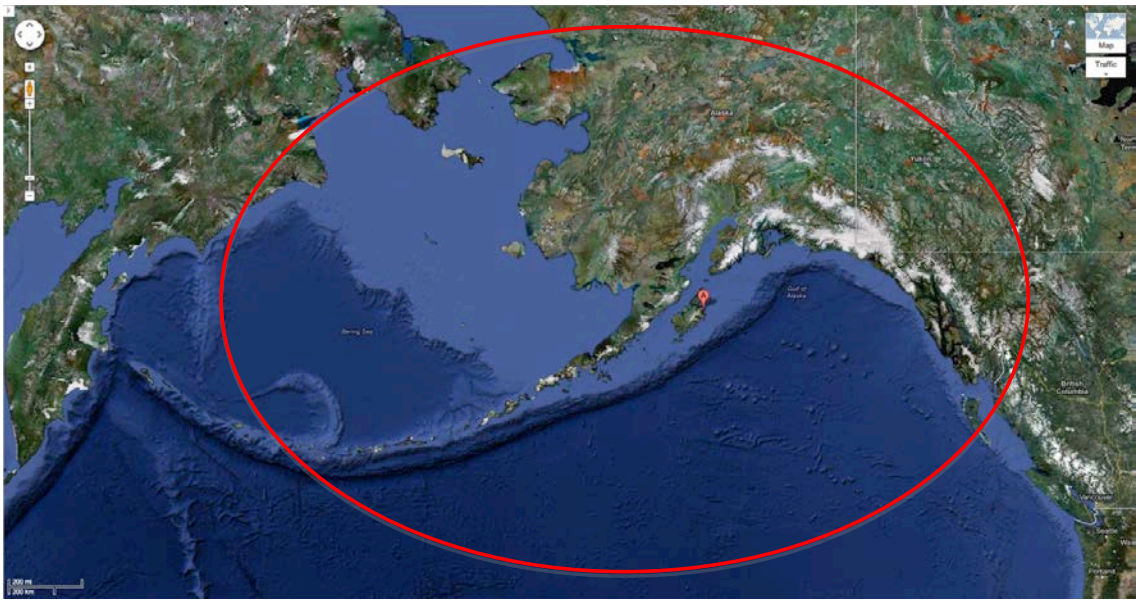

Figure 24. Conceptual SAR Operating Radius (Google Maps, 2013)

Costs for developing the interfaces were assigned to each system, as well as a cost for operating the system for a month in the case of the ISR or for 3 days in the case of the SAR. The deadline for development of an interface was assigned a value of: 0 – ready now, 1 – will be ready by the end of this epoch, or 2 – won't be ready this epoch, but the next. You may spend interface funds on an interface that won't be ready until the next epoch, but you get no performance increment from that interface in this epoch. An overall 'relative' performance value was assigned to each system based on its key capability. The costs for development were rough figures similar to what appear in official and informal budgetary estimates for interfacing with communications systems and integrating the mission systems to be able to interoperate. The costs to operate aircraft or other systems were determined similarly, in units of thousands of dollars per hour.

A Model Building Basis for SAR

New tools are being developed that could make the integration of the SoS exploration and analysis tools developed here even easier to use. When we built the SAR model, we looked into auto generating the domain input data from a more general model of the system. It does appear possible, but some development would have to be done. The activity diagram in Figure 25 built using classes that equate to the types of systems we used in this effort, is an example of the way that today's architects are being taught. This is the way they've been trained to think and communicate architecture concepts to others. This relatively new tool can already auto generate an execution timeline shown in Figure 26. Multiple executions can be set up in Monte Carlo simulations to obtain analysis statistics as well. This is the type of connection between tools that might be fruitful to pursue in future work.

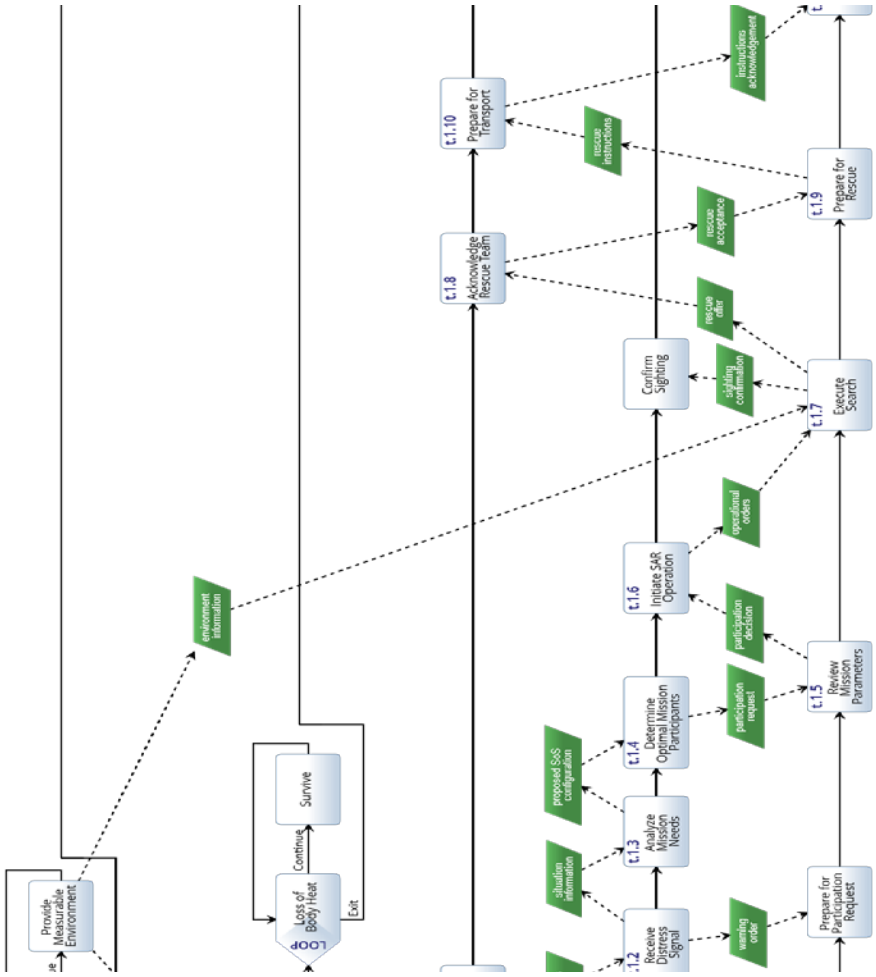

Figure 25. Activity diagram matching the CONOPS of the SAR model

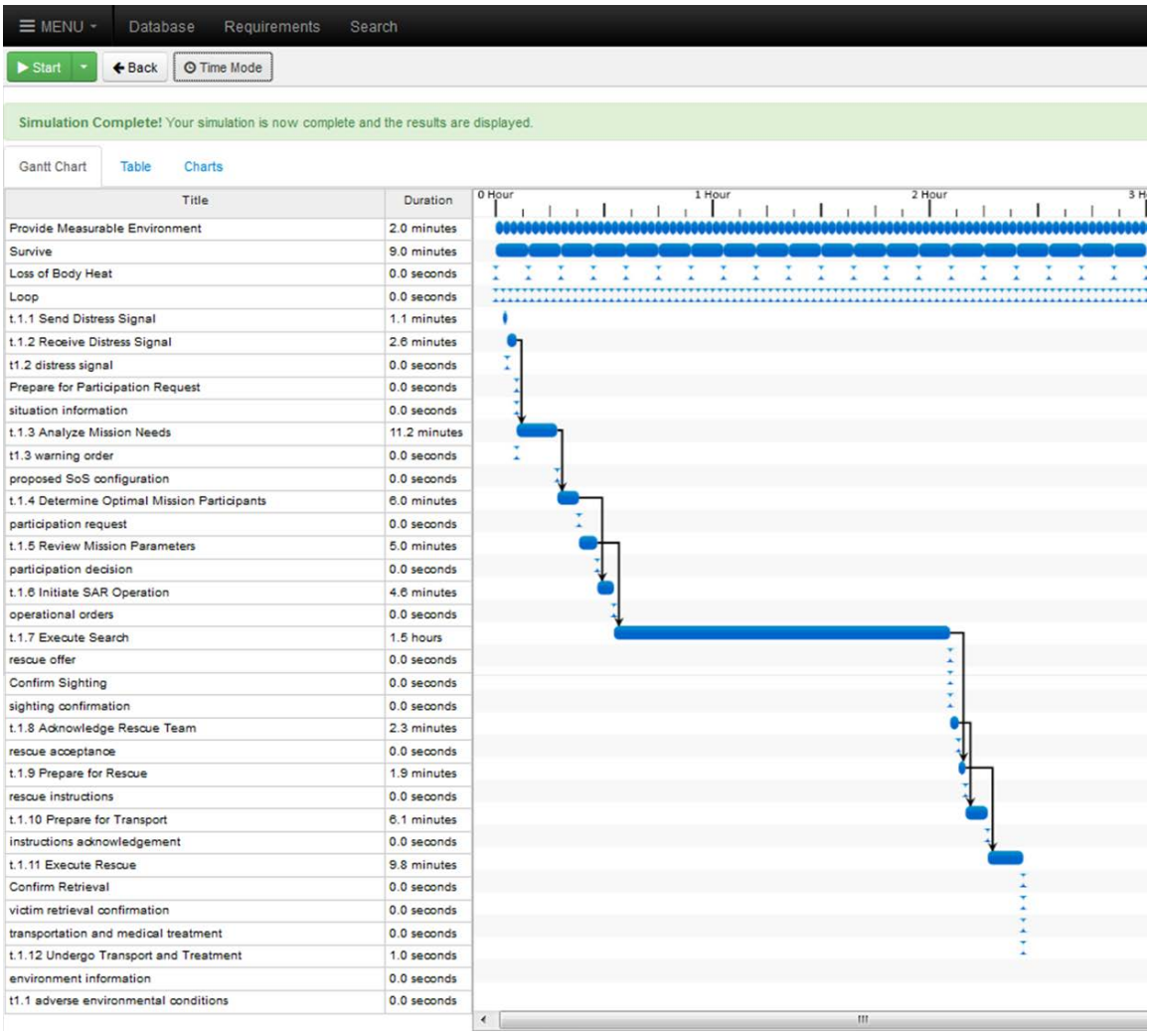

Figure 26. . Execution timeline example for SAR model

4.2.4 Mapping Attribute Measures to Fuzzy Variables

The generic membership function discussed, section 3.2.2 must be mapped to the values of the domain specific SoS. In a real problem, this mapping of ranges for estimated values for cost estimates vs. budget, or performance in terms of square miles searched per hour or tons of freight delivered would come from the problem and the stakeholders. The probability of success, or the number of shipments would have desired thresholds that would define the unacceptable, marginal, acceptable, or exceeds expectations grade for each attribute. Since we were attempting to explore the entire meta-architecture space with formulation, we came up with a backwards method of defining the membership functions.

After assigning the rough performance, cost, and capability mapping to individual systems selected, we created random chromosomes with the varying bias toward ones through the evaluation algorithms to

see where the possible values came out. Two example random but biased populations of 25 chromosomes are shown in Figures 27 and 28 for the 22 system ISR SoS. Explanations of these charts are presented in Table 10:

Table 10. Explanation of value exploring graph pages

1) On the first row of graphs, the number of ones in the whole chromosome (in blue) and five times the number of systems in the chromosome (in red) plotted together on the same scale
2) The overall sos evaluation on the 1 to 4 scale of unacceptable to exceeds expectations
3) The performance of each of the chromosomes, with dashed lines of different colors representing the edges of the membership functions
4) The flexibility attribute evaluation of each chromosome
5) On the second row of graphs, the maximum loss in performance by successive individual system removal of each participating system – that is, the robustness attribute
6) The value of the penalty/reward function for using infeasible/feasible interfaces for each chromosome
7) The total cost for each chromosome, and
8) The affordability, which is the total cost modified by (one minus the bump) raised to the penalty/reward power, as desribed in section above.

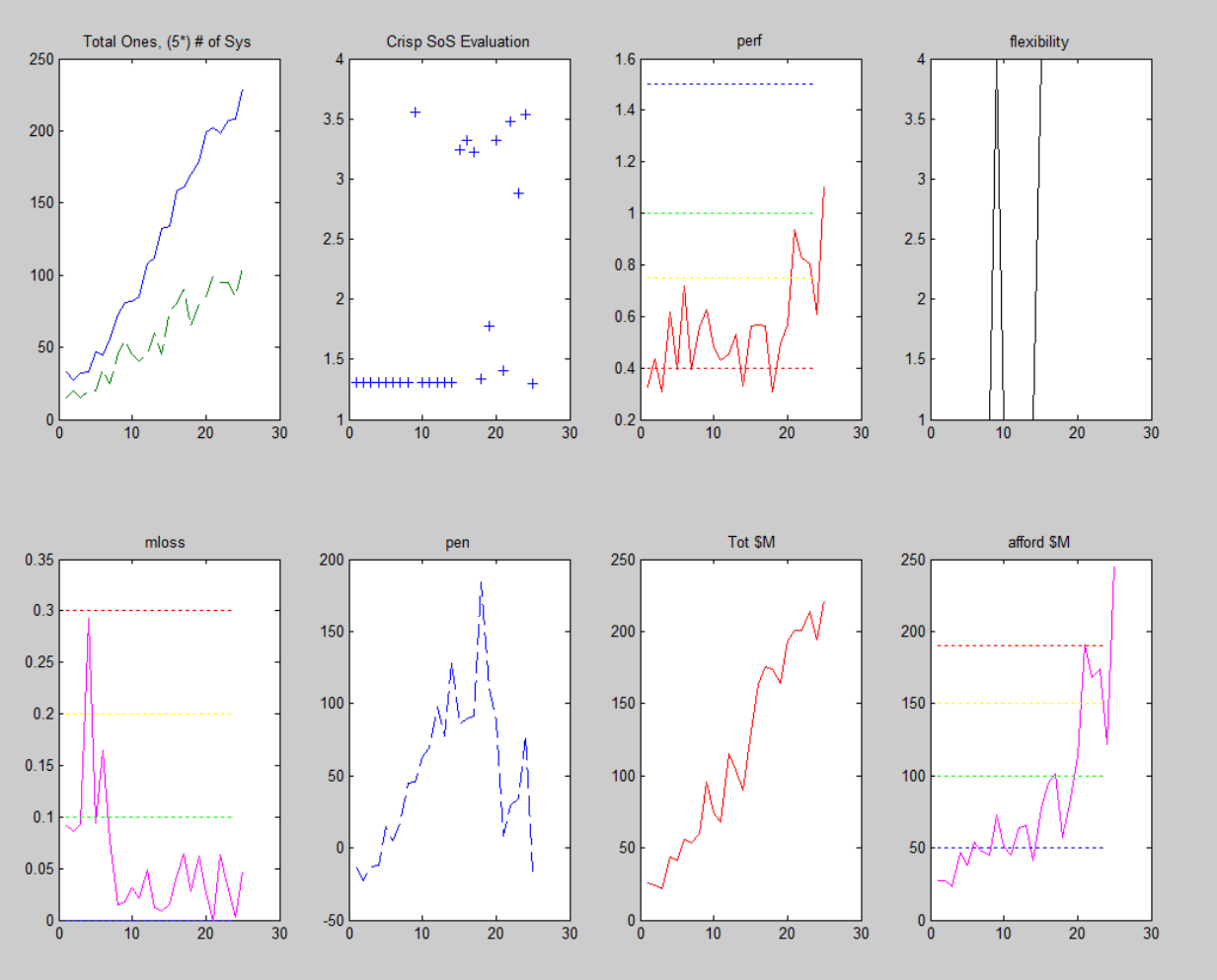

Figure 27. Exploring the meta-architecture - 25 chromosomes, example 1

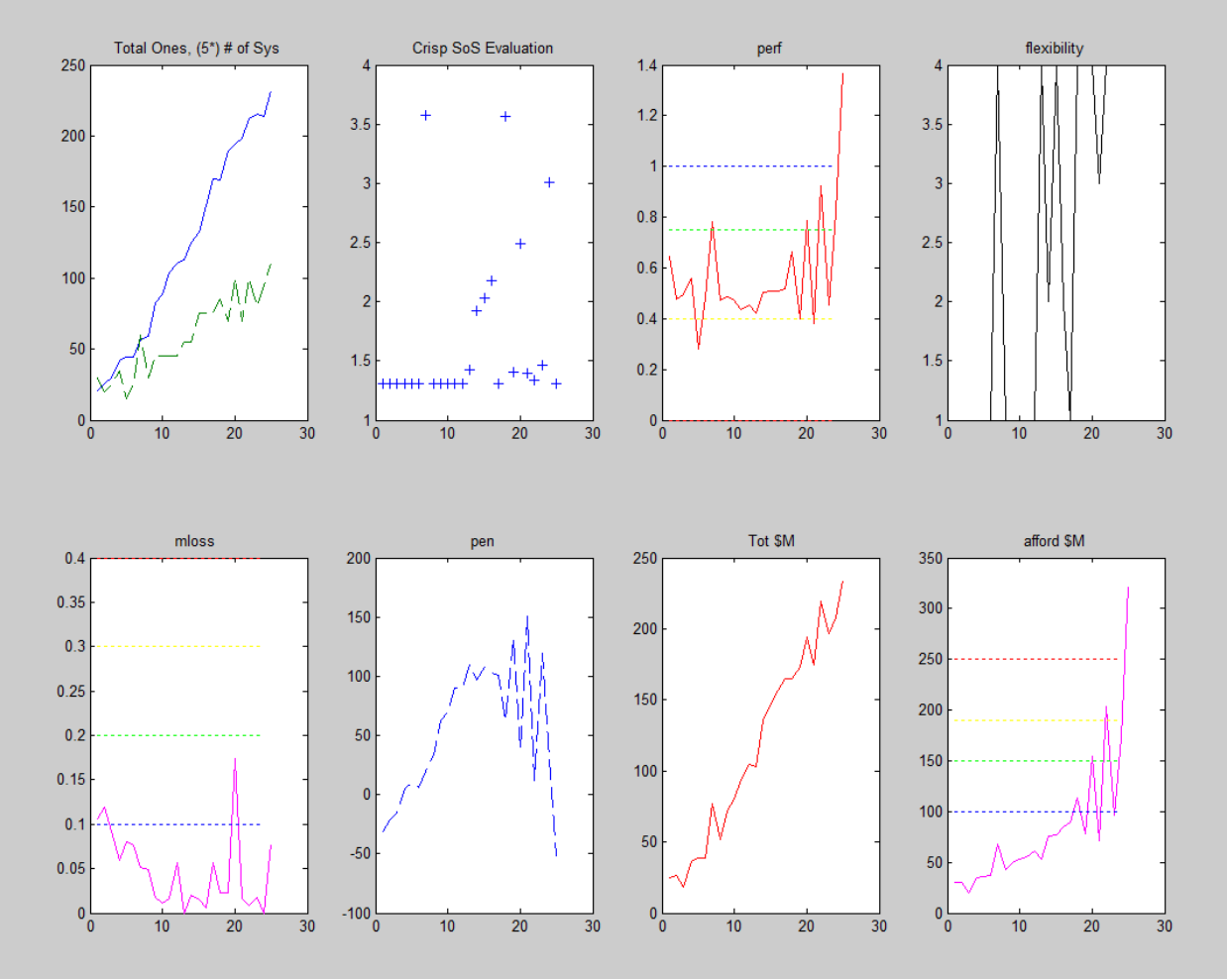

Figure 28. Exploring the meta-architecture to map membership function edges, Example 2

By running a few thousand random chromosomes (with the biased number of systems and interfaces) through the fuzzy evaluation subroutine, one can settle on adequate values for the membership function edges to show there are good solutions possible within the model, shown in Figure 28. This is not the 'finding the best chromosome' process, but only setting the membership functions so we can be sure of finding *some* acceptable chromosomes during the GA from which to select better mutations from each generation. You can also see similar shapes of the functions for each of the attributes and the penalty function on Figures 29 and 30. One important feature is that tiny changes in the chromosome can have wide swings in the values of each of the attributes. The search for a 'good' chromosome is really that, a search for it – it is not obvious where it is from the build up of the model.

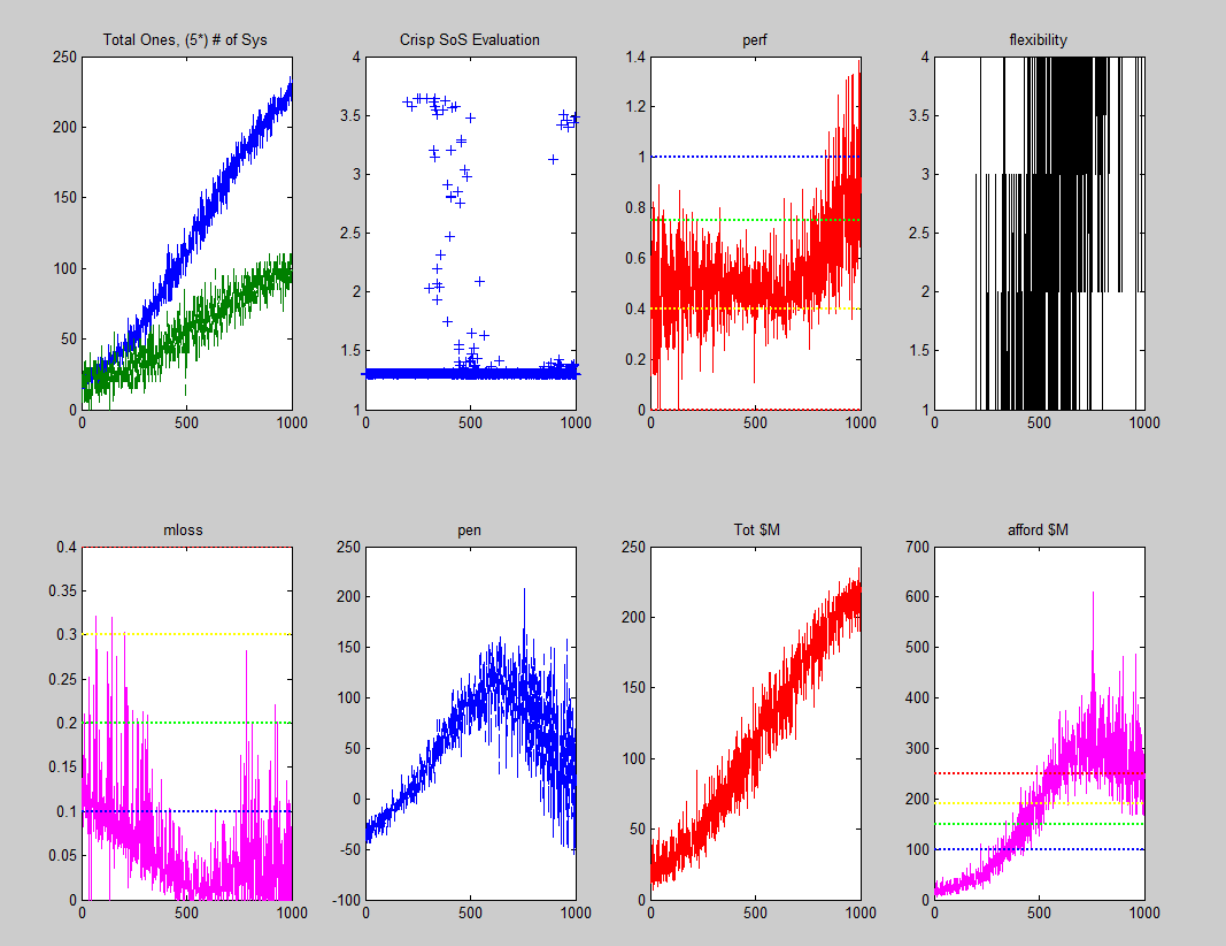

Figure 29. Setting the membership function edges for the attributes

It takes only a few minutes to run 1000 'almost' random chromosomes through the exploration phase. Some iterations on selection of the mapping values for the boundaries between membership functions may be required. If one selects values that are too tight, such as with a high performance, the robustness limits may need to be adjusted. The ISR membership function mapping is shown in When the membership function edges, the input domain specific costs and performances, and the limits for the robustness function are selected so that there are at least some chromosomes that are performing well, the next step is to run the full fuzzy model through the GA for 60 to 100 generations, as discussed in section 4.2. Minor kinks in the mapping lines show that the slopes of the membership function maps do not need to be constant as shown in Figure 30.

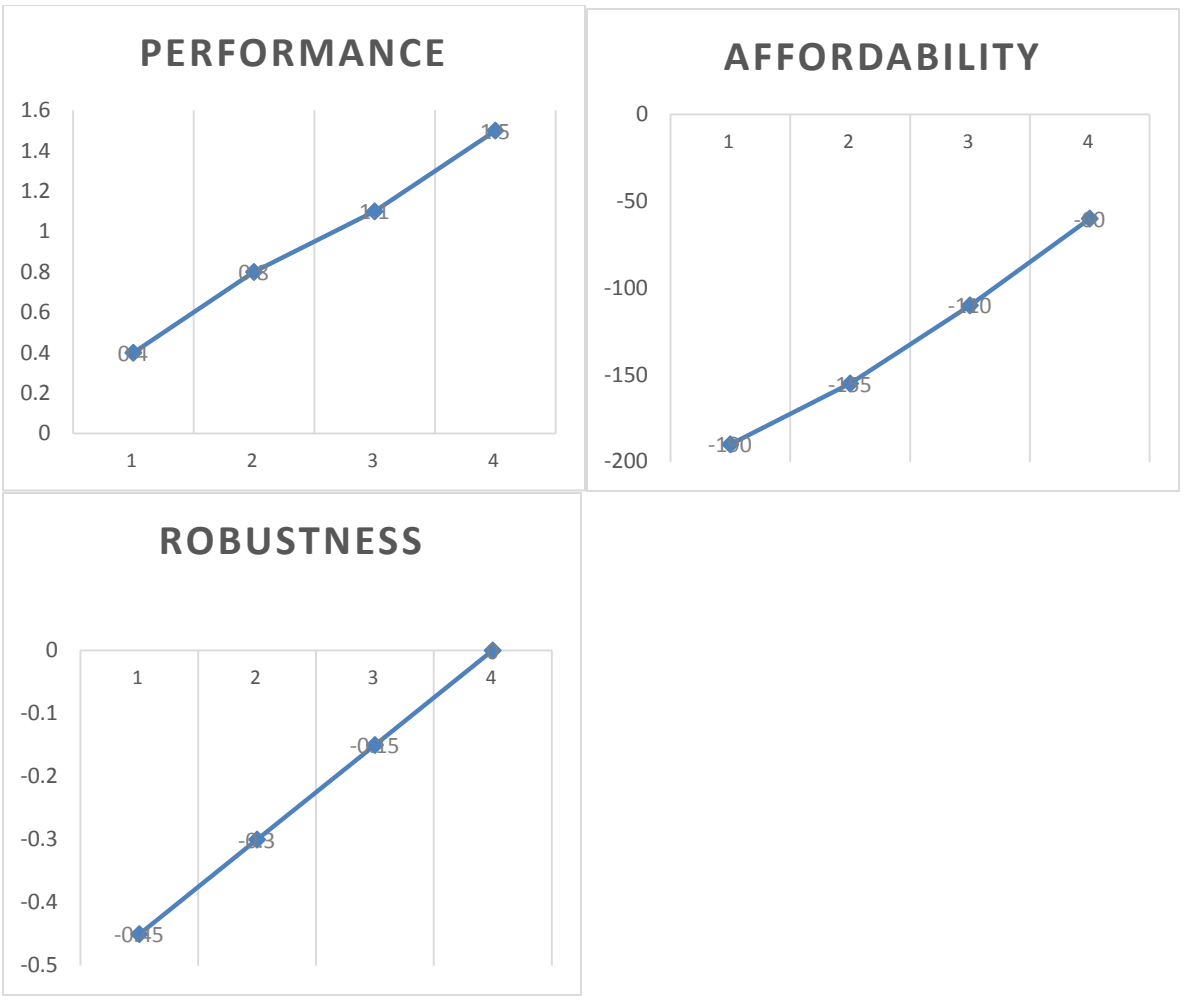

Figure 30. Attribute Values, Mapped to Fuzzy Variables

4.2.5 Rules For Combining Attribute Valuations To An SoS Evaluation

Performance Attribute:

Performance – for the ISR Domain example is made up of surveillance coverage in area per hour and wavelength region, combined with ability to reach the site of a discovered but fleeting high value target before it disappears.

- Background Assumptions: 100,000 square miles in which to hide; 30 minutes from start to finish for an operational launch; about 60 TELs operational; an individual TEL might hide for several days, so the probability of an individual TEL popping out to make a launch is only about 10% per day.

Rules for combining capabilities into performance:

- Fighters can provide modest capability in non-traditional ISR with on board sensors, *and* deliver several weapons, but they cost more to operate than many other systems
- Remote Piloted Aircraft (RPAs) can provide better ISR capabilities with somewhat less speed and single weapon capabilities, but also require a control station for each 2 RPAs. They are considerably cheaper to operate than fighters
- JSTARS can provide considerable radar ISR, and LOS and BLOS relay, but no weapons
- DSP can provide reliable notice of an actual launch, which means there definitely was a TEL in the open at that launch point, but it is not very precise localization of that point, meaning some search is still required upon an armed vehicle's arrival in the vicinity, and it takes a couple minutes to receive the data from DSP. The TEL can hide quickly after launch, leaving not much time to arrive there, find and attack it before disappears again. In the performance model, I multiplied the DSP coverage by 0.01 to account for the likely lack of closure from a DSP detection
- U-2 or Satellite can cover large area with high resolution, but turnaround time is hours; participation of U-2 or Satellite effectively decreases total area to be searched by other ISR platforms by a reasonable percentage, but does not affect real time surveillance. DSP is basically free, because it is used for other purposes
- Area to be covered is divided into sectors among the participating surveillance systems
- Time to arrive is proportional to the square root of the sector area being covered by each type of system, plus some time for transmitting data to, and double checking by, the exploit systems to insure the target is valid and not in a restricted area
- Probability of successful engagement is *defined* as 50% if coverage is the total area in half an hour, and time to arrive after detection is less than 10 minutes. Fighters or RPAs making the discovery are able to attack relatively quickly, transit time is less than 5 min for fighters, 10 min for RPAs; other types of detection require transit time for the attack vehicle which may be longer if it is in a different sector.

It would be possible to write fuzzy rules for evaluating some or all of the SoS attributes, as well. Gegov develops this concept in a discussion of fuzzy networks. [161]

4.2.6 ISR Domain Model Detail

The ISR Domain Model contains the following types of Systems, with existing capabilities, cost bogeys for development and operation, and likely development times as follows:

- Fighter aircraft with ElectroOptic/Infrared (EO/IR) pods,
- Fighters with Synthetic Aperture Radar (SAR),
- U-2 aircraft and Satellites with EO/IR,
- DSP missile launch detection satellites,
- JSTARS aircraft with SAR, 6) Remotely Piloted Aircraft (RPA),

- Line of sight (LOS), that is, tactical; and Beyond Line of Sight (BLOS) data links,
- Exploitation centers in theater as well as in Continental United States (CONUS) for analyzing ISR data, and
- Control stations for the RPAs.

The five types of capabilities ($SOS.C_j$) provided by Systems and interfaces listed below are how we build up the SoS capabilities:

- EO/IR
- SAR
- Communications links
- Exploitation Centers
- Control Centers

4.3 SoS Negotiation Model Implemented

The current SoS negotiation model can be describes as follows:

If the change in performance required by SoS and offered by system agent is less than 50% or the change in funding offered by the SoS to acquire the capability and demanded by system agent is greater than 50%, SoS decides to drop the system from architecture. Similarly, if the system agent does not agree to participate in the first wave itself, SoS concludes the negotiation. These thresholds on funding, performance, and deadline can be justified as all changes above or below the prescribed limits ,as the case may be, lead to an unacceptable SoS architecture quality. If the individual system request is above these threshold values than SoS manager can start negotiation.

Moreover if the systems acquiesce to the requirements of the SoS agent in one epoch of negation and SoS agent finds the overall architecture quality above the predefined limit set by the acknowledged SoS manager, it concludes the first wave of the ABM model.

Else, if the SoS architecture quality is below the cutoff value SoS starts a fresh round of negotiations (or the second epoch) with the system agents. These negotiations continue until SoS agent achieves the cutoff value of the architecture quality. Consequently if the SoS agent is not able to form an architecture god enough to meet the user requirements and the predefined number of negotiations has exceeded, SoS agent ends the negotiation and decides the to generate another meta-architecture. This meta-architecture is generated to explore the trade space, as the negotiations are not leading to a possibility within the cutoff values. This process is still in the first wave of ABM.

Once a wave is concluded, the SoS agent saves the final architecture chromosome as a starting point for the next wave of the architecture evolution.

4.4 ISR Implementation of Fuzzy Genetic Optimization Model

Genetic algorithms (GA) have been used to solve the optimization model for the meta-architecture, both for suggesting a starting chromosome, and for selecting negotiation solutions between the SoS and system agents.

4.4.1 Representation Of Meta–Architectures In Genetic Algorithm Format

The chromosomes of the population *p* in one generation of the GA were represented as a matrix of the linear chromosome *m(m+1)/2* bits wide by *p* rows deep. Each row of the population was evaluated for the attributes (determined by the domain input data and the chromosome) and the overall fitness of that chromosome (determined by the fuzzy assessment of the four attributes). The rows were sorted by the SoS fitness, and the best 20% (with the addition of one chromosome from each quintile) were selected to reproduce to the next generation. This guaranteed a small fraction of very different chromosomes than the top few percent were always included for reproduction. This helps insure the process cannot get "stuck" on a purely local optimum.

4.4.2 Ensuring Meta-Architecture Generated Provides Desired System Capabilities

The first generation of chromosomes is generated almost randomly except for a changing bias toward ones from first to last in the population. This is similar to the way we explored the meta-architecture space to set the membership function size, described in section 4.2.4. This insured that the starting tournament did not have a narrow range of the number of ones in the chromosome, that is an 'average' number of systems and an average number of interfaces with small variations around the mean that would occur if the population was filled in with equally likely ones and zeroes. Even though we tried creating the population chromosome set randomly, we found this did not "seed" the full range of the meta architecture space. Instead, by biasing each member of the initial population toward an average likelihood of ones roughly equal to the ordinal position of that chromosome divided by p, a very wide variation in number of systems and interfaces is ensured. Subsequent generations are generated from the top 20% of the population in fitness (tournament winners), through the genetic operators defined below. The best chromosome from each generation is always kept intact.

To speed convergence, we biased the communication system participation to 60% in the initial population. This was not necessary, but it changed the required number of generations from many hundred to less than one hundred consistently. This was accomplished by changing the (still random) likelihood of a 1 in the last four columns in the upper triangular representation to 60%.

4.4.3 Genetic Operators

The following genetic operators were used in the GA implementation [81]. The top 20% in fitness is replicated 4 more times to fill out the next generation population, then the following genetic operations are performed on each 20% of the population.

- Mutation: a random number is called from a flat distribution for each bit in the chromosome. If the number is less than a settable percentage, typically 0.5% to 2%, (we experimented with rates up to 10%) then the bit was reversed. A second 20% of the next generation had a doubled mutation rate.

- Crossover: a random number from a flat distribution is called for each chromosome; this random number is multiplied by total bits in the chromosome. Two new chromosomes are created from two old chromosomes by taking the first part of one cut at the random bit just selected and combining it with the second part from the next chromosome, and vice versa. This is done in two fifths of the population. This method is also known as sexual reproduction.
- Transposition: in the last fifth of the population, a portion of the chromosome is transposed. The length is random up to 10 times the mutation percentage of the length of the whole chromosome. That makes it an average of about 13 bits, with a range from 2 to 25 bits for our 22 system SoS. The transposed section is cut out, the following part of the chromosome moved left, and the snipped section is spliced back in the chromosome its own length from the original starting point. This is similar to crossover (above), but only a short section of the chromosome is transposed, and only one parent is used to make the offspring chromosome.

The mutation operator makes relatively small changes to the chromosome, but the crossover and transposition can make very large changes as well as relatively small ones. This keeps the distribution of bits well mixed in the next generation of chromosomes, so that a significant portion of the meta-architecture is explored.

4.4.4 Key performance attributes identification and selection

The key SoS attributes were identified by the panel of subject matter experts and stakeholders in section 3.2.3. Trial definitions and algorithms for evaluating each attribute, which depended on the binary chromosome representation, were experimented with until we reached a set of algorithms and capability estimates for each system with differentiation among a range of trial chromosomes that appeared reasonable to stakeholders.

In the generic sense, the method does not change if there are many performance (or other SoS attributes) values to be evaluated. The functional form of the performance attribute used here was primarily a summation of individual system contributions of area searched per hour, with token values assigned to non-search systems' capabilities such as command and control or communications. The first attempt at this had many dependencies and non-linearities (including speed and time on station), but explaining the nuances to stakeholders was just too difficult. The second attempt was much simpler and approximate, giving only a rough approximation of one systems' contribution compared to the other systems, but still showed the major effects of different numbers of systems of various types and capabilities.

4.4.5 Data and File Structures Used in Integrating the Models to Generic ABM

1. Generating Capability, Performance, Cost, Schedule Matrices

After evolving and selecting a chromosome based on SoS fitness, the information to provide the individual systems for the negotiation phase is prepared. An Excel file with 4 sub matrices equivalent to (SOS.M, and SOS.C_g) for capability allocated to systems and interfaces, SOS.P for performance to each capability, SOS.f for funding, and SOS.d for deadlines, is prepared in the format of file

"domainParameters.xlsx." This is created from the domain specific model coded in the Matlab file "DomainModel.m" for the GA to process into individual system files for the SoS agent to provide to each system agent.

2. Individual System Data Outputs

An output file is created for each system S_j involved in the SoS architecture where $j = 1, 2, \ldots m$. The file is constituted of a table of multiple rows. Each row depicts one capability C_i where $i = 1, 2, \ldots n$. This includes whether the system S_j has been assigned to the capability C_i. If assigned, s_{ij} shows the value '1' indicating assignment of system S_j to capability C_i. Otherwise the value shown is '0'. Similarly, the row shows the assignment status of all the system interfaces r_{ik}. In addition, it provides the aggregate capability performance, deadline and funding for each capability C_i.

Figure 31 below shows the partial Excel file table for individual systems specifying capability participation of the system and interfaces in up to ten various capabilities are used. The deadline, funding and performance for each capability are also provided. More specifically the figure shows part of an output sheet for system 1. A complete spreadsheet includes all the respective interfaces. There are 22 files that are provided as output.

n= 5 capabilities
m= 22 systems
j= 1

	Architecture (the portion related to system j)					
	S_j	$I_{j,1}$	$I_{j,2}$	$I_{j,3}$	$I_{j,4}$	$I_{j,5}$
C_1	0	0	0	0	0	0
C_2	0	0	0	0	0	0
C_3	1	0	0	1	1	0
C_4	1	0	0	0	0	0
C_5	0	0	0	0	0	0

Figure 31. Individual System Capability Participation

3. SoS Data Output

An output file of the system of system A_{SoS} architecture is created . Each cell depicts a system S_j has been selected to be part of SoS architecture. If selected, S_j shows the value '1'. Otherwise the value shown is '0'.

Figure 32 depicts the output including 22 systems and respective interfaces totaling 253 systems and interfaces. The sheet has been truncated for visibility.

Systems Contributing to the SoS Architecture
1 means system is contributing
0 means system is not contributing

	Systems						
	S_1	S_2	S_3	S_4	S_5	S_6	S_7
Final Architecture	1	0	1	1	1	1	1

Figure 32. Output chromosome

4. Domain Parameter Data Inputs

An Excel file with four tables specifying the domain parameters required for the genetic algorithms process is used.

I. Figure 33 specifies the assignment s_{ij} for each system S_j, and the assignment r_{ik} for each interface I_k. This is provided for each capability C_i. The following truncated sheet clarifies.
II. Figure 34 specifies the performance $\mathrm{p_{ij}}$ for each system $\mathrm{S_j}$, and the performance $\mathrm{p_{ik}}$ for each interface $\mathrm{I_k}$. This is provided for each capability $\mathrm{C_i}$. The following truncated sheet clarifies.
III. Similarly the third table (not shown) depicts the funding $\mathrm{f_{ij}}$ and $\mathrm{f_{ik}}$ for each system and interface respectively.
IV. Finally the last table (not shown) provides the deadline values $\mathrm{d_{ij}}$ and $\mathrm{d_{ik}}$ for each system and interface respectively.

DOMAIN
PARAMETERS
m= 22 systems
n= 5 capabilities

Capability Provided by Systems 1 means capability provided by system 0 means capability not provided by system

	Systems										
	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}
C_1	0	0	1	1	0	1	0	0	0	0	0
C_2	0	0	1	1	1	1	1	0	0	0	0
C_3	1	0	1	1	0	1	0	0	0	0	1
C_4	1	0	0	0	0	1	0	0	0	0	0
C_5	0	1	0	1	1	1	0	1	1	1	1

Figure 33. Capabilities

System Performance for Each Capability

	Systems						
	S_1	S_2	S_3	S_4	S_5	S_6	S_7
C_1	0	0	5	2	0	2	0
C_2	0	0	5	5	1	2	1
C_3	4	0	7	3	0	10	0
C_4	3	0	0	0	0	3	0
C_5	0	5	0	3	3	3	0

Figure 34. Performance

4.5 ISR Implementation of System Negotiation Models (Parameter)

This phase reports the parameter values chosen throughout the ABM model to formulate ISR domain problem.

Number of systems is =22; Number of capabilities possessed =5; the fuzzy assessor used to assess the fitness of architectures has 4 key performance attributes as inputs. The KPP'S are performance, affordability, flexibility and robustness. The values for the edges of the membership functions are given in Table 11.

The negative numbers signify that better is smaller, so that all maps go to the right from poorer to better values.

Table 11. Boundary of the membership mapping from real values to fuzzy membership functions

Attributes	*mapfuzlow*	*Fuzzy Value 1*	*Fuzzy Value 2*	*Fuzzy Value 3*	*Fuzzy Value 4*
Performance – Pk per day for SoS	0	0.4	0.7	1	1.5
Affordability - $M cost	-250	-190	-150	-110	-60
Flexibility	0	1	2	3	4
Robustness – Loss of performance	-0.75	-0.45	-0.3	-0.15	0

The SoS architecture quality threshold is fixed at 3.1

4.5.1 Selfish Negotiation Model parameters

Throughput (units of capability produced per time unit per resource unit) is a vector of n elements is represented by the symbol y here."$y(i) = 0$" means the individual system k is not capable of providing $capability\ i$. y is a nonnegative real value vector. The greater $the\ y(i)$, the more efficient the $system\ k\ in\ building\ capability\ i$. The values for vector y currently used in the model are given below.

y	4.5	4.5	4.5	4.5	4.5

Resource per time unit (assuming a symmetric distribution of resource) is given by two variables namely Z_avg and Z_range. The values given to Z_avg =10 and Z_range=1

Costs

The cost vector is defined as consuming one unit of resource per time unit in producing capabilities. It is a vector with of n elements. "c_c(i)= 0" if "y(i)=0" .

c_c	0.1	0.1	0.1	0.1	0.1

The cost of consuming one unit of resource per time unit in producing interfaces, as a percentage of c_c. It is a vector of n elements. "c_I(i) = 0" if "y(i)=0"

c_I	5.00%	5.00%	5.00%	5.00%	5.00%

The continuous growth rate per time unit is taken = 0.5%
The required profit margin of capabilities is denoted by pm and it describes the minimum required rate of return from providing capabilities. It is a vector of n elements. "$pm(i) = 0$" if "$y(i) = 0$"

pm	20%	20%	20%	20%	20%

Negotiation Parameters

(P1) tells the best performance of the system k at the given deadlines and funding.

(P2) tells the additional funding needed and additional time needed in order to meet the performance requirements (if the system k is capable).

In (P2) the additional funding needed may be greater than the minimum additional funding needed; similarly, the additional time needed may be longer than the minimum additional time needed.

That is, CriticalPro is the probability of sending the result of (P1) to SoS, and (1-CriticalPro) is the probability of sending the result of (P2) to the SoS. There are two extreme scenarios:

$$If: CriticalPro = 0, the\ result\ of\ (P2)\ is\ sent\ to\ SoS;$$

$$If: CriticalPro = 1, the\ result\ of\ (P1)\ is\ sent\ to\ SoS;$$

CriticalPro=0.5

Funding

If the system k will use up all funding to be provided by SoS, it requests up to AF1 more than the additional funding needed. AF1= 10%

If the system k will not use up all funding to be provided by SoS, it requests up to AF2 more than the funding provided by the SoS.

AF2=5%

Deadline

the system k request up to AT units of time more than the minimum additional time needed

AT=1

4.5.2 Opportunistic Negotiation Model parameters

Three levels of each factor were created and combination of the values was used as negotiation parameters.

$l = \ system\ behavior\ factor; m = \ system\ behavior\ factor; eta = budget\ estimating\ factor$

Test	l	m	eta
1	1.1	1.1	0.1
2	1.5	1.1	0.1
3	1.9	1.1	0.1
4	1.1	1.6	0.1
5	1.5	1.6	0.1
6	1.9	1.6	0.1
7	1.1	2	0.1
8	1.5	2	0.1
9	1.9	2	0.1
10	1.1	1.1	0.5
11	1.5	1.1	0.5
12	1.9	1.1	0.5
13	1.1	1.1	0.9
14	1.5	1.1	0.9
15	1.9	1.1	0.9
16	1.1	1.6	0.5
17	1.5	1.6	0.5
18	1.9	1.6	0.5
19	1.1	1.6	0.9
20	1.5	1.6	0.9
21	1.9	1.6	0.9
22	1.1	2	0.5
23	1.5	2	0.5
24	1.9	2	0.5
25	1.1	2	0.9
26	1.5	2	0.9
27	1.9	2	0.9

4.5.3 Cooperative Negotiation Model parameters

N=5 capabilities

M= 3 issues under negotiation

S=22 number of systems

The minimum and maximum values of the attributes estimated by the individual systems are given below:

$(min_1, max_1) = (10, 60) sq.\ miles$--performance

$(min_2, max_2) = (10, 50) \$M$--funding

$(min_3, max_3) = (1, 3) wave\ cycle$--deadline

$V = 0.5$; V varies for each individual system depending on his cooperativeness.

$$W2 = f^{e^{\alpha(\#\ of\ epochs - 1)}} \text{ where } \alpha = 0.1$$

$$Total\ no\ of\ epochs\ (negotiation\ cycles)\ estimated\ to\ be = 5$$

Maximum number of interfaces possible is denoted by I_{Max}=21

$$Normalized\ weights\ for\ trade\ off\ between\ the\ two\ innate\ behaviors\ \ w1 = 0.7, w2 = 0.3$$

5 Results of first wave of ABM Acknowledged SoS for sample ISR Problem

5.1 Results of Fuzzy Genetic Optimization of Sos Meta-Architecture

The meta-architecture generated by the fuzzy genetic approach is presented below. The architecture quality along with the values for the key performance attributes on a scale of 1-4 is given below.

	Arch Quality
Architecture	**3.7365**
Performance	2.737108999
Affordability	3.296602926
Flexibility	3
Robustness	3.310474933

The meta-architecture generated has an overall quality of 3.7365.

System Numbers	Capabilities possessed
System 1- System 7	(1 and 5)
System 8- System 9	(1) only
System 10- System 13	(2 and 5)
System 14- System 16	(3 and 5)
System 17- System 18	(4 and 5)
System 19- System 22	(5) only

Systems and interfaces selected in the meta-architecture are represented as binary zeros and ones.

S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	S_{16}	S_{17}	S_{18}	S_{19}	S_{20}	S_{21}	S_{22}
1	1	0	0	0	0	1	1	0	0	1	1	1	1	0	0	0	1	0	0	1	1

Interfaces to System 1																				Interfaces to System 2																			
$I_{1,2}$	$I_{1,3}$	$I_{1,4}$	$I_{1,5}$	$I_{1,6}$	$I_{1,7}$	$I_{1,8}$	$I_{1,9}$	$I_{1,10}$	$I_{1,11}$	$I_{1,12}$	$I_{1,13}$	$I_{1,14}$	$I_{1,15}$	$I_{1,16}$	$I_{1,17}$	$I_{1,18}$	$I_{1,19}$	$I_{1,20}$	$I_{1,21}$	$I_{1,22}$	$I_{2,3}$	$I_{2,4}$	$I_{2,5}$	$I_{2,6}$	$I_{2,7}$	$I_{2,8}$	$I_{2,9}$	$I_{2,10}$	$I_{2,11}$	$I_{2,12}$	$I_{2,13}$	$I_{2,14}$	$I_{2,15}$	$I_{2,16}$	$I_{2,17}$	$I_{2,18}$	$I_{2,19}$	$I_{2,20}$	$I_{2,21}$	$I_{2,22}$
0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	1	0	0	1	0	0

Interfaces to System 3																				Interfaces to System 4																			
$I_{3,4}$	$I_{3,5}$	$I_{3,6}$	$I_{3,7}$	$I_{3,8}$	$I_{3,9}$	$I_{3,10}$	$I_{3,11}$	$I_{3,12}$	$I_{3,13}$	$I_{3,14}$	$I_{3,15}$	$I_{3,16}$	$I_{3,17}$	$I_{3,18}$	$I_{3,19}$	$I_{3,20}$	$I_{3,21}$	$I_{3,22}$	$I_{4,5}$	$I_{4,6}$	$I_{4,7}$	$I_{4,8}$	$I_{4,9}$	$I_{4,10}$	$I_{4,11}$	$I_{4,12}$	$I_{4,13}$	$I_{4,14}$	$I_{4,15}$	$I_{4,16}$	$I_{4,17}$	$I_{4,18}$	$I_{4,19}$	$I_{4,20}$	$I_{4,21}$	$I_{4,22}$
0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

Interfaces to System 5																	Interfaces to System 6																	
$I_{5,6}$	$I_{5,7}$	$I_{5,8}$	$I_{5,9}$	$I_{5,10}$	$I_{5,11}$	$I_{5,12}$	$I_{5,13}$	$I_{5,14}$	$I_{5,15}$	$I_{5,16}$	$I_{5,17}$	$I_{5,18}$	$I_{5,19}$	$I_{5,20}$	$I_{5,21}$	$I_{5,22}$	$I_{6,7}$	$I_{6,8}$	$I_{6,9}$	$I_{6,10}$	$I_{6,11}$	$I_{6,12}$	$I_{6,13}$	$I_{6,14}$	$I_{6,15}$	$I_{6,16}$	$I_{6,17}$	$I_{6,18}$	$I_{6,19}$	$I_{6,20}$	$I_{6,21}$	$I_{6,22}$
1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0

Interfaces to System 7															Interfaces to System 8													
$I_{7,8}$	$I_{7,9}$	$I_{7,10}$	$I_{7,11}$	$I_{7,12}$	$I_{7,13}$	$I_{7,14}$	$I_{7,15}$	$I_{7,16}$	$I_{7,17}$	$I_{7,18}$	$I_{7,19}$	$I_{7,20}$	$I_{7,21}$	$I_{7,22}$	$I_{8,9}$	$I_{8,10}$	$I_{8,11}$	$I_{8,12}$	$I_{8,13}$	$I_{8,14}$	$I_{8,15}$	$I_{8,16}$	$I_{8,17}$	$I_{8,18}$	$I_{8,19}$	$I_{8,20}$	$I_{8,21}$	$I_{8,22}$
0	0	0	0	1	0	1	0	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1

Interfaces to System 9													Interfaces to System 10											
$I_{9,10}$	$I_{9,11}$	$I_{9,12}$	$I_{9,13}$	$I_{9,14}$	$I_{9,15}$	$I_{9,16}$	$I_{9,17}$	$I_{9,18}$	$I_{9,19}$	$I_{9,20}$	$I_{9,21}$	$I_{9,22}$	$I_{10,11}$	$I_{10,12}$	$I_{10,13}$	$I_{10,14}$	$I_{10,15}$	$I_{10,16}$	$I_{10,17}$	$I_{10,18}$	$I_{10,19}$	$I_{10,20}$	$I_{10,21}$	$I_{10,22}$
0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	1	0	0	0	0	0	1	0	0

Interfaces to System 11											Interfaces to System 12									
$I_{11,12}$	$I_{11,13}$	$I_{11,14}$	$I_{11,15}$	$I_{11,16}$	$I_{11,17}$	$I_{11,18}$	$I_{11,19}$	$I_{11,20}$	$I_{11,21}$	$I_{11,22}$	$I_{12,13}$	$I_{12,14}$	$I_{12,15}$	$I_{12,16}$	$I_{12,17}$	$I_{12,18}$	$I_{12,19}$	$I_{12,20}$	$I_{12,21}$	$I_{12,22}$
0	0	1	0	1	0	1	0	0	1	1	0	0	0	0	0	0	0	0	1	0

Interfaces to System 13									Interfaces to System 14								Interfaces to System 15							Interfaces to System 16						
$I_{13,14}$	$I_{13,15}$	$I_{13,16}$	$I_{13,17}$	$I_{13,18}$	$I_{13,19}$	$I_{13,20}$	$I_{13,21}$	$I_{13,22}$	$I_{14,15}$	$I_{14,16}$	$I_{14,17}$	$I_{14,18}$	$I_{14,19}$	$I_{14,20}$	$I_{14,21}$	$I_{14,22}$	$I_{15,16}$	$I_{15,17}$	$I_{15,18}$	$I_{15,19}$	$I_{15,20}$	$I_{15,21}$	$I_{15,22}$	$I_{16,17}$	$I_{16,18}$	$I_{16,19}$	$I_{16,20}$	$I_{16,21}$	$I_{16,22}$
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0

Interfaces to System 17					Interfaces to System 18				Interfaces to System 19			Interfaces to System 20		I to S21
$I_{17,18}$	$I_{17,19}$	$I_{17,20}$	$I_{17,21}$	$I_{17,22}$	$I_{18,19}$	$I_{18,20}$	$I_{18,21}$	$I_{18,22}$	$I_{19,20}$	$I_{19,21}$	$I_{19,22}$	$I_{20,21}$	$I_{20,22}$	$I_{21,22}$
0	0	0	1	0	0	0	1	0	1	0	1	0	0	0

Three sets of results are obtained by assuming all participating system follow the same negotiation model each time.

$$System.S_i = \{cooperative, opportunistic, selfish\}$$
$$System.Information_i = f(\Delta d, \Delta f, \Delta p)$$

Table 12 is a representation of the generic form of outputs or response of the individual system to the offer made by SoS.

Table 12 Outputs of the negotiation

Architecture (the portion related to system j)																							Deadline	funding	performance
S_j	$I_{j,1}$	$I_{j,2}$	$I_{j,3}$	$I_{j,4}$	$I_{j,5}$	$I_{j,6}$	$I_{j,7}$	$I_{j,8}$	$I_{j,9}$	$I_{j,10}$	$I_{j,11}$	$I_{j,12}$	$I_{j,13}$	$I_{j,14}$	$I_{j,15}$	$I_{j,16}$	$I_{j,17}$	$I_{j,18}$	$I_{j,19}$	$I_{j,20}$	$I_{j,21}$	$I_{j,22}$	Sysemj.Δd_i	Sustemj. Δf_i	Sysemj.Δp_i
1	**1**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
1	**0**	**0**	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**0**	**1**	**0**	**1**	**1**	**1**	**1**	**1**	**1**			
1	**1**	**0**	**0**	**0**	**0**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**0**	**0**	**1**	**0**	**0**			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

5.1.1 Generational Results for the ISR GA with Fuzzy Fitness and Attribute Assessments

The format for interpreting the GA process eight-chart compilation graphics (Figure 35) is shown in Figures 35-42.

Table 13. Explanation of GA graph pages

1) On the first row of graphs, the number of ones in the whole chromosome (in blue) and five times the number of systems in the chromosome (in red) plotted together on the same scale. These are not in generation order, but sorted by the fitness, from worst to best, so in early generations this is disorderly. In later generations it shows the result of the mutation operators.
2) The overall SoS evaluation on the 1 to 4 scale of 1 = unacceptable to 4 = exceeds expectations; the small number in the upper left is the generation number
3) The performance fuzzy variable of each of the chromosomes (in black), and the flexibility fuzzy variable (in blue)
4) A histogram of the crisp value of the SoS evaluation for this generation's population
5) On the second row of graphs, the fuzzy value of the robustness attribute
6) The affordability fuzzy value for each chromosome (scale is 1 to 4)
7) The total cost for each chromosome, just as a cross check on the fuzzy value, and
8) The upper triangular display of the generation's best chromosome ones and zeroes with the feasibility color coded: Black is an unused (zero) and infeasible position; Blue is an unused but feasible interface – the performance could be better if it were used; Red is a used (has a one in it) interface that is INfeasible (either one system not there, or no communication path between systems) – penalized; and green is a used and feasible interface or system – rewarded. The numbers in the lower left are the SoS evaluation and the number of each color for feasibility penalty/reward. The system type labels are on the right margin.

The GA output displays are presented from Figure 35 through Figure 38. Figure 35 is a typical initial population of the GA. It shows the widely varying number of ones in the population of chromosomes and the generally poor SoS performance as well as attribute values for a series of random selected chromosomes. The next two figures show an intermediate generation, and the Figure 38 shows the final solution. Figures 39 shows the improvement in the fitness of the SoS by generation; the top line is the best fitness, the second line is fitness of the 20th percentile chromosome. The top line, showing the SoS evaluation, should ratchet toward the final solution, never growing smaller; the fitness value of the 20th percentile depends on the genetic operators, so it may dip from one generation to the next.

This ISR GA model is currently implemented to provide input to the ABM model.

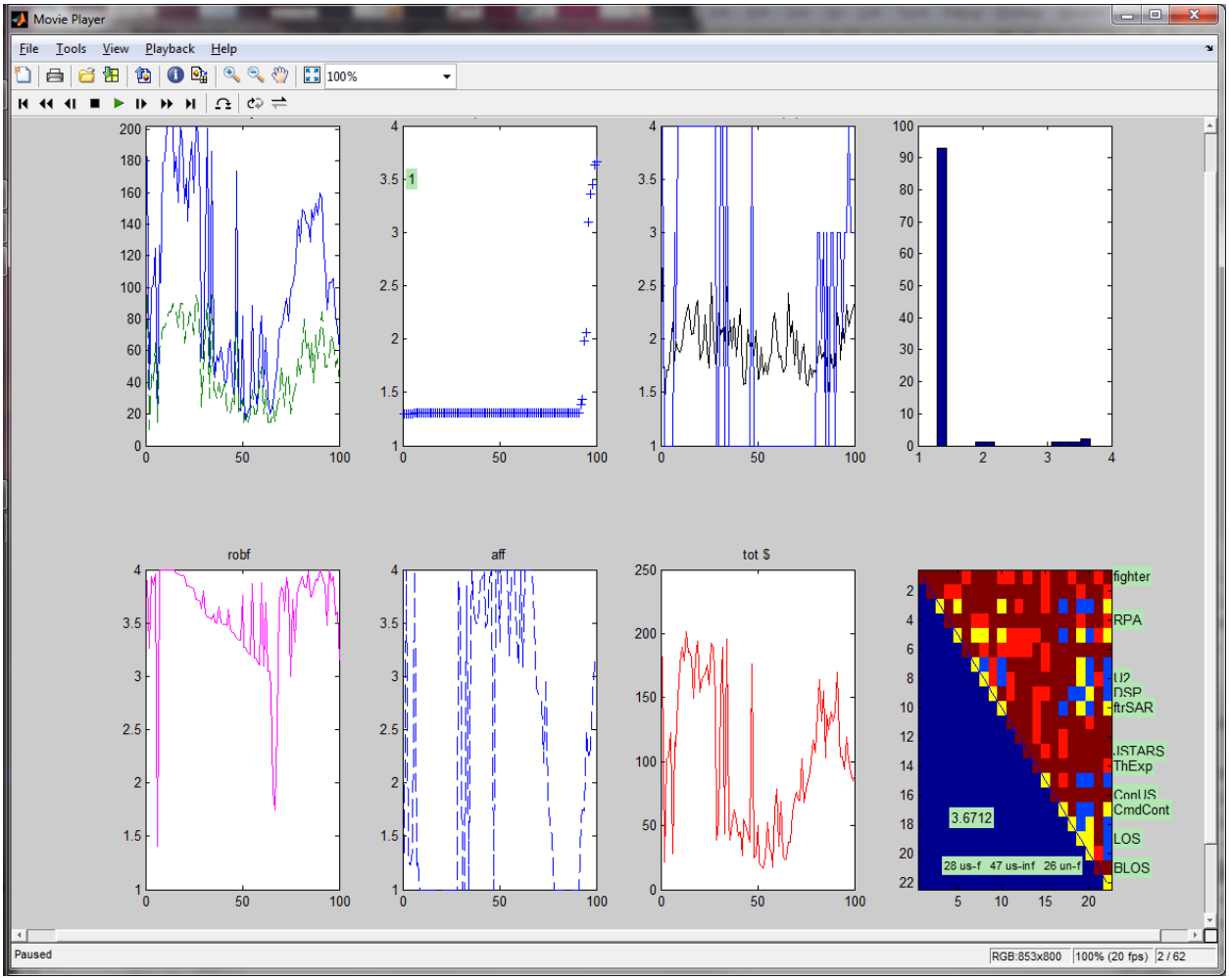

Figure 35. Attribute evaluations for initial population of ISR chromosomes in the GA; the SoS fuzzy values, and best chromosome of the generation displayed

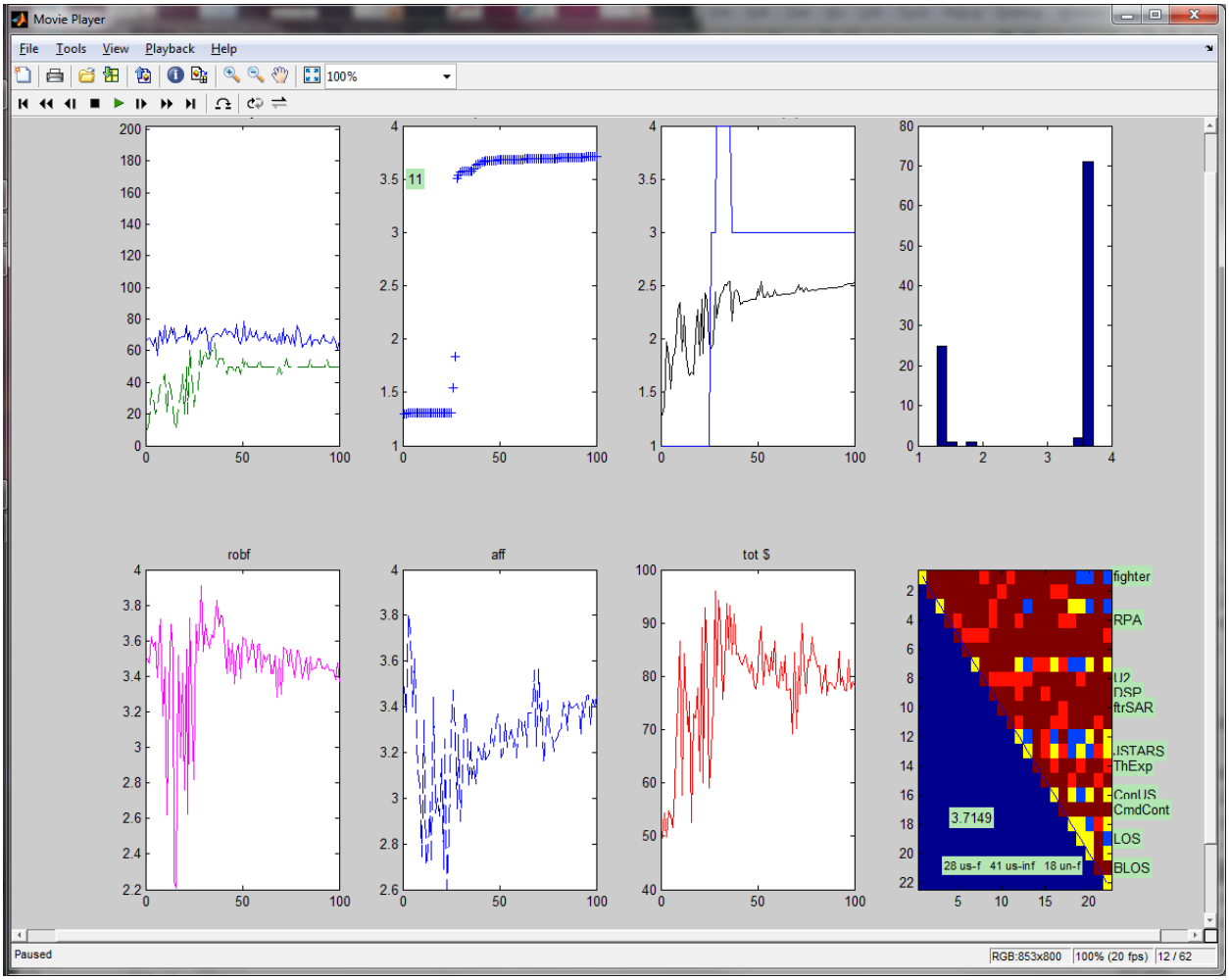

Figure 36. Intermediate generation from the GA; all population values sorted by SoS fitness

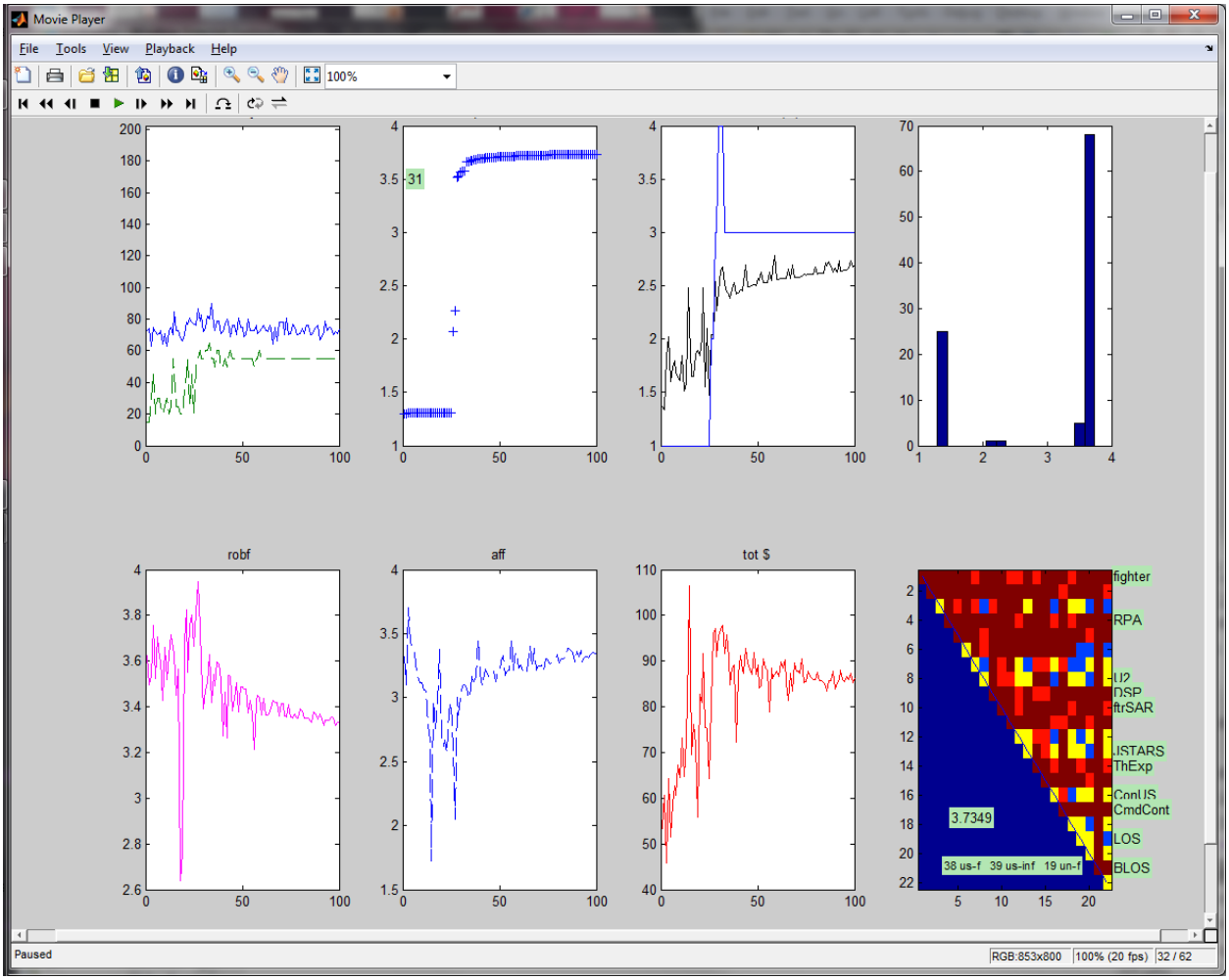

Figure 37. 31st generation; note changes in system participation, interfaces, and best fitness value

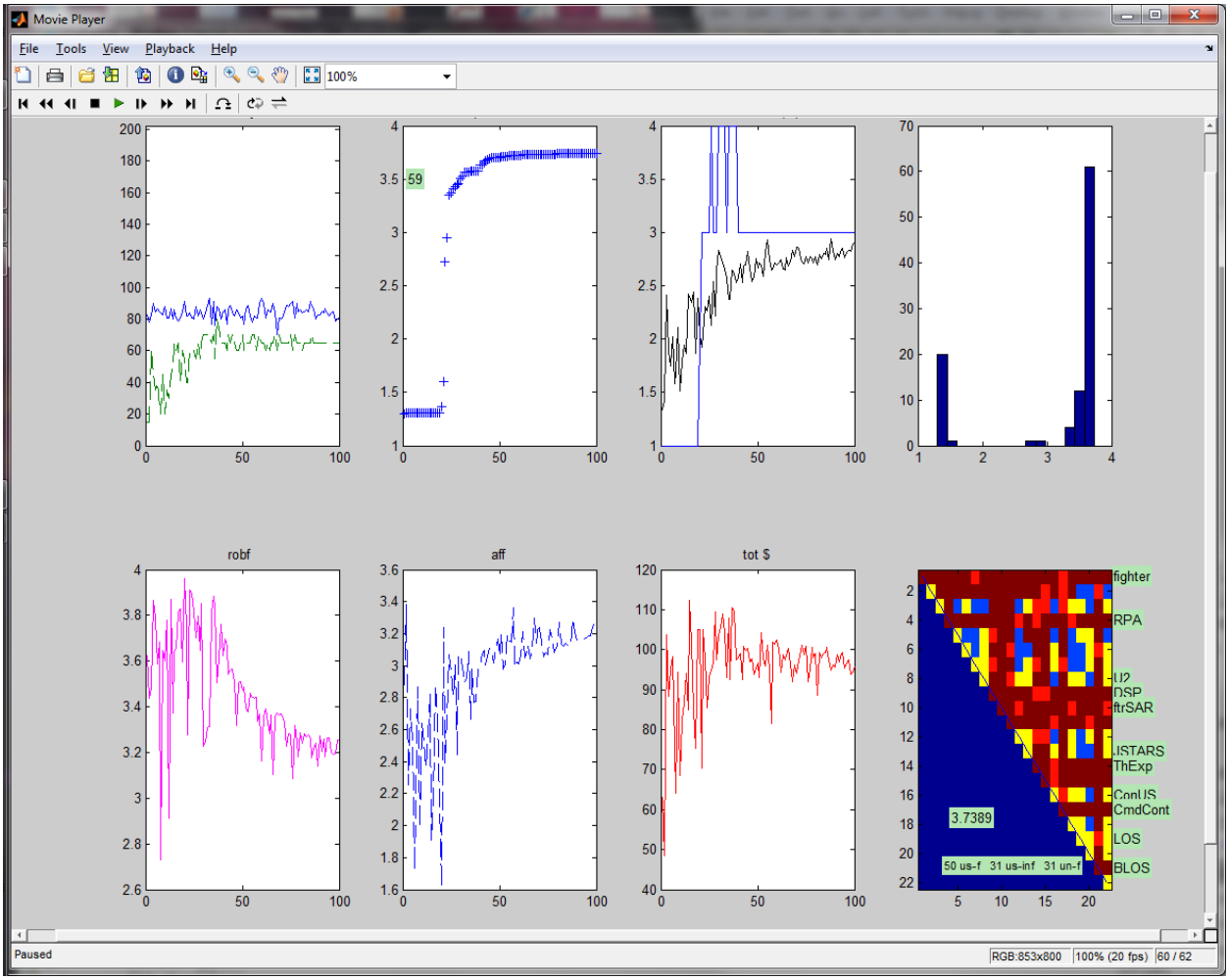

Figure 38. The last generation of the GA for this ISR run

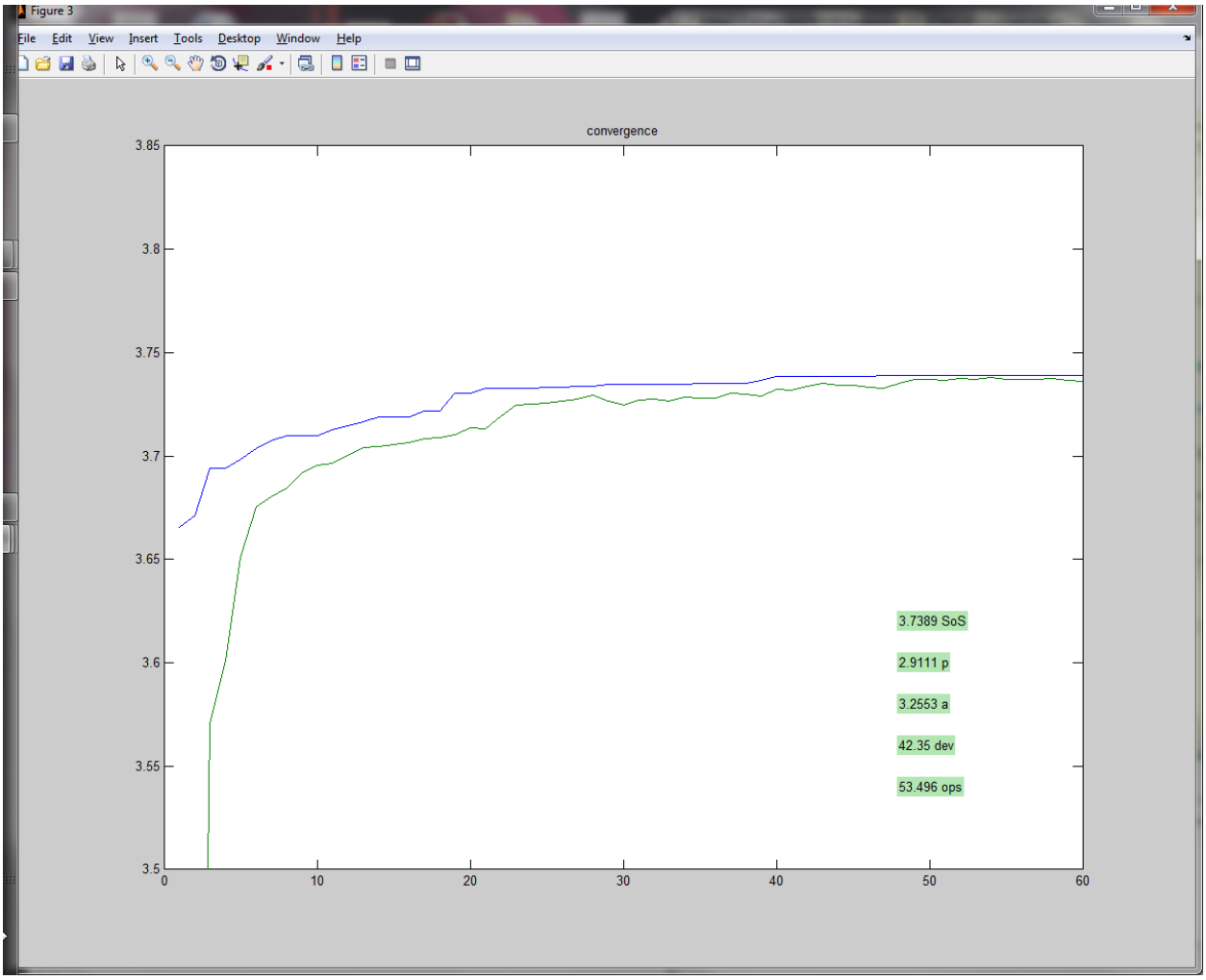

Figure 39. Typical, and classic, generational convergence of the GA with the blue line; green line is the 20th percentile chromosome

5.1.2 GA Process Results for SAR model

The next four figures (Figure 40 through Figure 43) show the same style of GA results for the SAR model. Note that there are 29 systems instead of 22 from the ISR model, and the names of the types of systems along the right side of the chromosome are different.

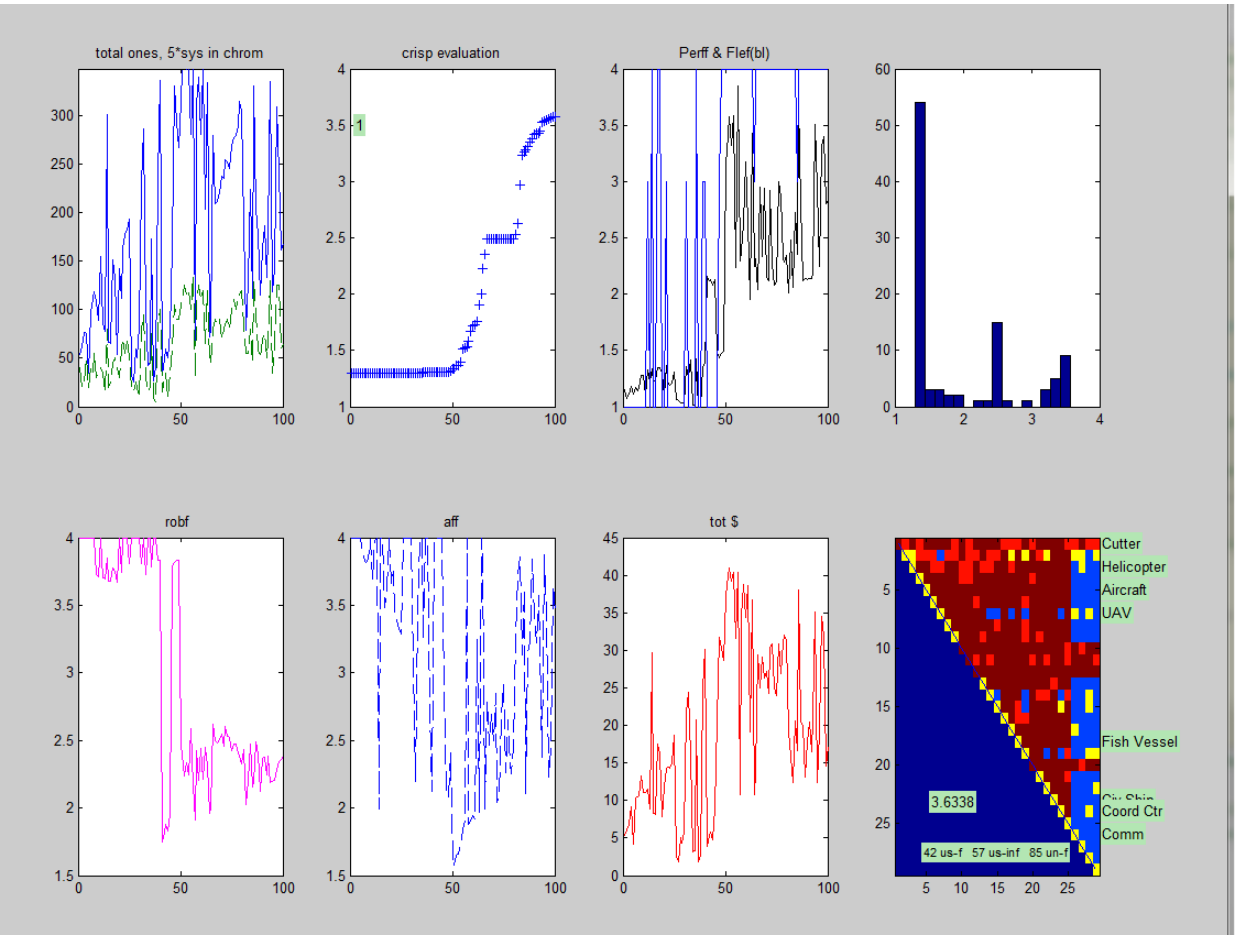

Figure 40. Initial population data plotted in the GA for SAR

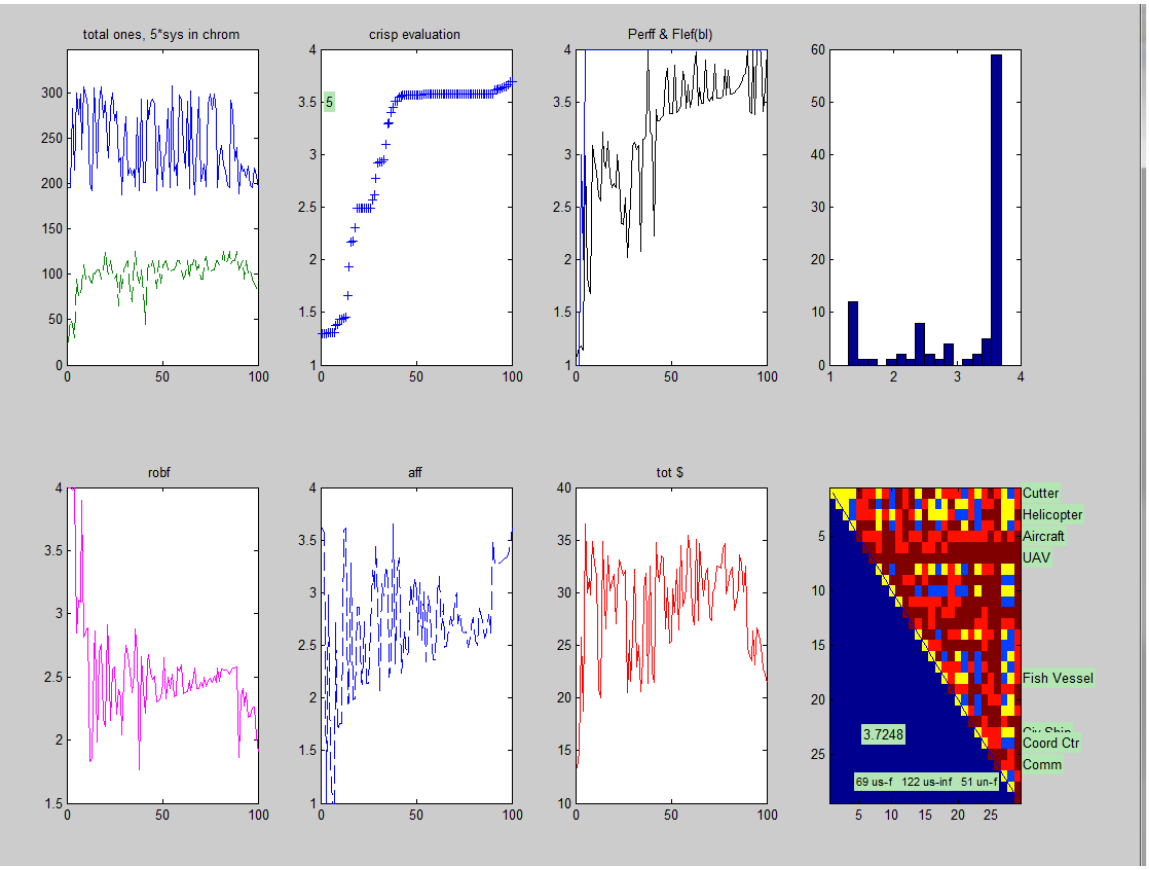

Figure 41. Generation 5 of the SAR GA

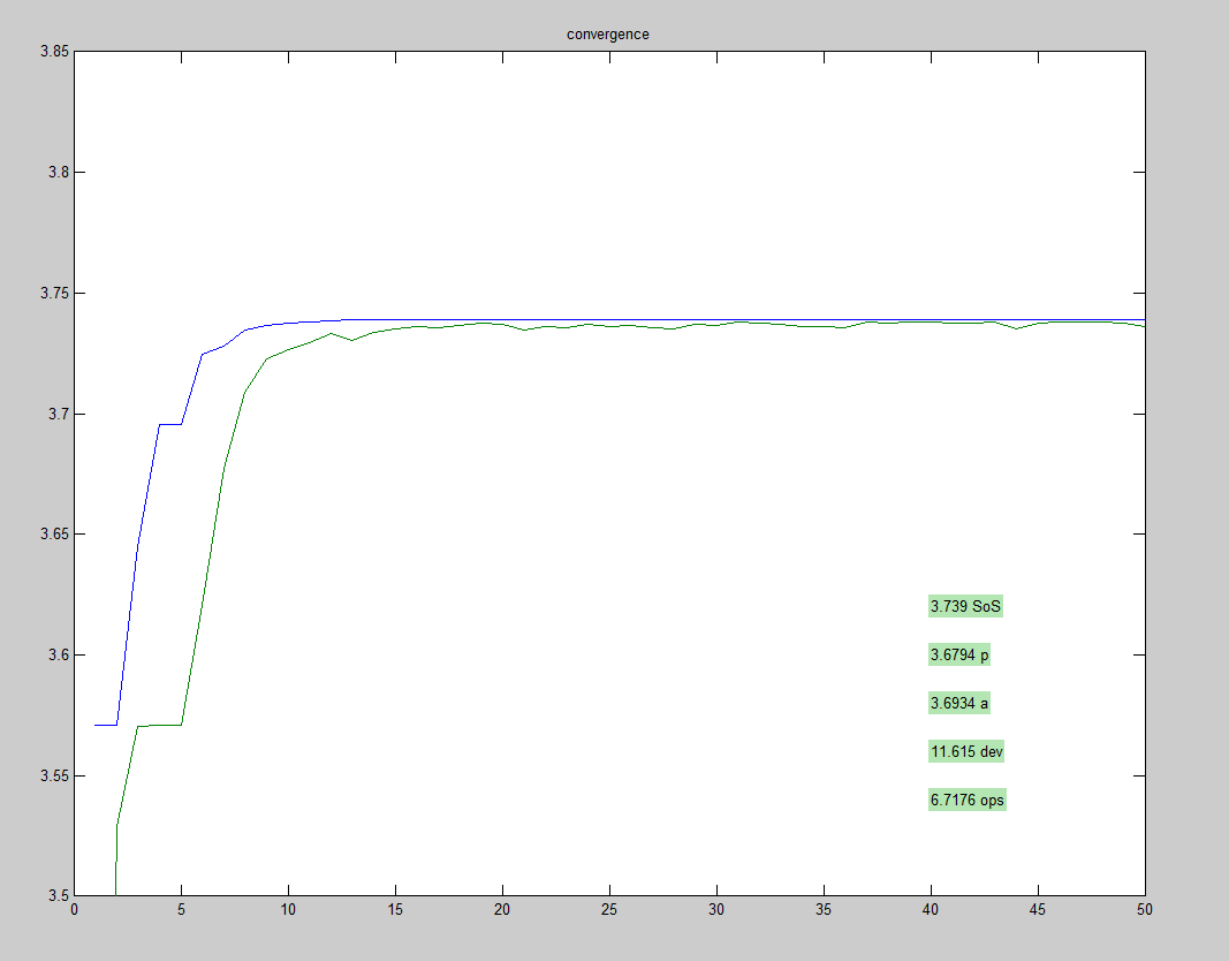

Figure 42. Convergence of the SAR GA model

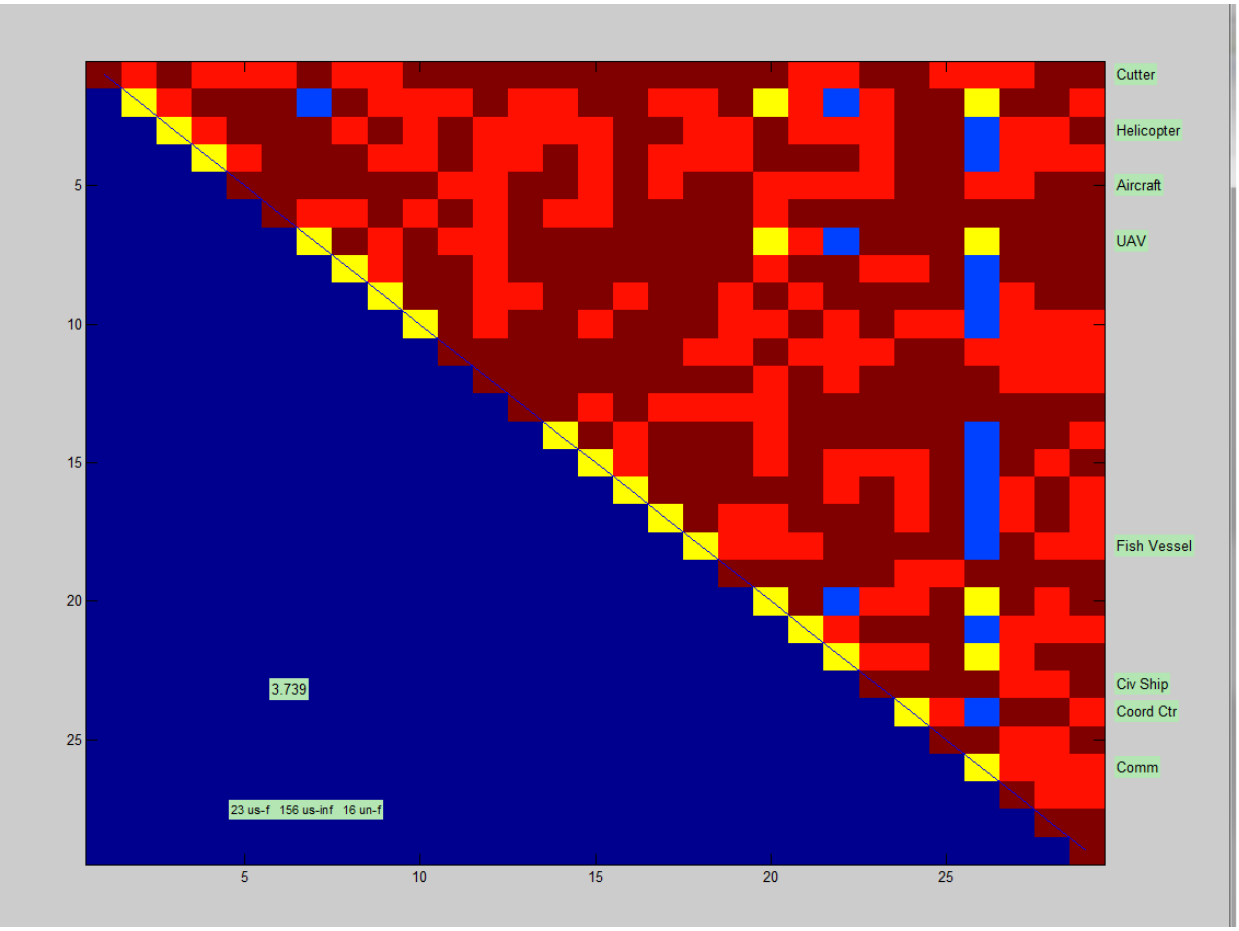

Figure 43. Final chromosome from the SAR GA model shows the impact of too few communication systems

5.1.3 Performance of the Fuzzy Assessor in the GA

It is interesting to note that both good and bad architectures appear across a range of number of participating systems and interfaces. This is true both for the overall SoS fitness, and for individual attributes scaled-to-fuzzy values. There is some correlation between performance and number of systems up to a point; there is generally no correlation between number of systems and affordability, with the other attributes being a bit more random.

5.1.4 Conclusions about The GA Model Approach

The implementation of GA in the above optimization is a one pass, non-dynamic approach. The trapezoidal membership functions have a relatively flat fitness output; the simple rule set does not lend itself to sharp distinctions from one chromosome to the next. Therefore, it is observed that the crisp fitness reaches a maximum early and does not sharply differentiate "near-by" chromosomes. There seem to be many wasted infeasible interfaces that use up funding without providing any benefit, and in

fact, detract from both performance and affordability. We could remap the membership values to a narrower region of the real world attribute values, then run the GA again to sharpen up the output. We could also investigate more intelligent gene specific GA operators. Nevertheless, we did find that even this simple implementation of the GA can relatively quickly find a reasonably good chromosome, that meets the intent of the selection process.

5.2 Selfish model results

The Table 14 illustrates the systems selected in the meta-architecture along with the corresponding capability they possess. This table helps the reader to understand the negotiations that proceed after the meta-architecture generation process. The Table 15 explains the offer made by the SoS agent based on the meta-architecture to the individual systems. The offer consists of three attributes namely funding, performance and deadline. These attributes are independent of each other. The funding is expressed in million dollars, performance in sq.km of area covered and deadline is measured as waves in the overall SoS architecting process. The concept of wave is similar to the concept of era in epoch-era analysis. Epoch-Era Analysis is an approach for describing systems over time as existing in a series of static contexts (epochs) that change stochastically. Many epochs constitute an era. The results of the first wave are presented in the following tables. Each table belongs to a particular system consisting of the values of three attributes that are demanded or provided by SoS. Each system as described above can provide only a certain set of capabilities; hence, the rows of capabilities, which cannot be provided by the particular system, are empty.

Table 14. System Selected and their Capabilities in an Meta-Architecture

Systems Selected for Negotiation (11)	Capabilities Possessed
System 1,2,7	(1 and 5)
System 8	(1) only
System 11,12, 13	(2 and 5)
System 14	(3 and 5)
System 18	(4 and 5)
System 21, 22	(5) only

Table 15. Selfish Negotiation Model results for System 1 (Accepted by SoS)

	Capabilities	Deadline $SoS.d_i$	Funding $SoS.f_i$	Performance $SoS.p_i$		Deadline $Systemj.\Delta d_i$	Funding $Sustemj.\Delta f_i$	Performance $Systemj.\Delta p_i$
The Offer by SoS to Sys 1	C1	1	10.2	13.5	System 1 response: Accepted by SoS to provide capabilities C1 and C5	1	0.48815514	0
	C2	0	0	0		0	0	0
	C3	0	0	0		0	0	0
	C4	0	0	0		0	0	0
	C5	1	10.55	16		0	0.25603565	0

Table 15 contains information regarding a bilateral negotiation between SoS and System 1. System 1 can provide Capability 1 (C1) and/or (C2) but no more. The values in the first row corresponding to C1 indicate that SoS requires System 1 to join in the first wave of the SoS architecture. SoS is providing a funding of 10.2 units to acquire C1 and demands performance level of 13.5 units for the same. The response to this offer of SoS can be read in the adjacent columns consisting of delta values of attributes.

The delta values correspond to $(delta = System\,response - SoS\,offer)$. Hence a positive value of "1"in deadline means that System is not ready to participate in the first wave but will be ready by the second wave. Equivalently the funding delta value is positive 0.488 meaning the System 1 has asked for an additional amount from SoS for providing C1. The Performance delta value is zero, which can be interpreted as the System 1 prepared to provide the required performance levels.

Similarly for C5, SoS asks the System 1 to join in the first wave, has provision of 10.55 units of funding to extract a performance of 16 units. The response is recorded as delta change in the adjacent columns. Based on the delta values of each attributes and the SoS negotiation thresholds, System 1 is accepted to be a part of the first wave of the architecting process. Table 16 through Table 25 present the results for the rest of the systems.

Table 16 Selfish Negotiation Model results for System 2 (Accepted by SoS)

	Capabilities	Deadline $SoS.d_i$	Funding $SoS.f_i$	Performance $SoS.p_i$		Deadline $Systemj.\Delta d_i$	Funding $Sustemj.\Delta f_i$	Performance $Systemj.\Delta p_i$
The Offer by SoS to Sys 2	C1	**1**	**10.1**	**11**	System 2 response: Accepted by SoS to provide capabilities C1 and C5	**0**	**0**	**-2.13163E-14**
	C2	1	0	0		0	0	0
	C3	**0**	**0**	**0**		**0**	**0**	**0**
	C4	**1**	**0**	**0**		**0**	**0**	**0**
	C5	1	10.2	12		0	0	-7.10543E-15

Table 17 Selfish Negotiation Model results for System 7 (Accepted by SoS)

	Capabilities	Deadline $SoS.d_i$	Funding $SoS.f_i$	Performance $SoS.p_i$		Deadline $Systemj.\Delta d_i$	Funding $Sustemj.\Delta f_i$	Performance $Systemj.\Delta p_i$
The Offer by SoS to Sys 7	C1	**1**	**2.2**	**13.5**	System 7 response: Accepted by SoS to provide capabilities C1 and C5	**1**	**0.08714281**	**0**
	C2	1	0	1		0	0	0
	C3	**1**	**0**	**0**		**0**	**0**	**0**
	C4	**1**	**0**	**0.5**		**0**	**0**	**0**
	C5	1	2.45	16		0	0.11753782	0

Table 18 Selfish Negotiation Model results for System 8 (Accepted by SoS)

The Offer by SoS to Sys 8	Capabilities	Deadline $SoS.d_i$	Funding $SoS.f_i$	Performance $SoS.p_i$	System 8 response: Accepted by SoS to provide capabilities C1 and C5	Deadline $Systemj.\Delta d_i$	Funding $Sustemj.\Delta f_i$	Performance $Systemj.\Delta p_i$
	C1	1	15	5		1	0.70049494	0
	C2	1	0	0.5		0	0	0
	C3	0	0	0.5		0	0	0
	C4	1	0	0		0	0	0
	C5	1	0.25	2		0	0	0

Table 19 Selfish Negotiation Model results for System 11 (Accepted by SoS)

The Offer by SoS to Sys 11	Capabilities	Deadline $SoS.d_i$	Funding $SoS.f_i$	Performance $SoS.p_i$	System 11 response: Accepted by SoS to provide capabilities C1 and C5	Deadline $Systemj.\Delta d_i$	Funding $Sustemj.\Delta f_i$	Performance $Systemj.\Delta p_i$
	C1	1	0	1.5		0	0	0
	C2	1	10.7	19.5		1	0.39757587	0
	C3	1	0	0.5		0	0	0
	C4	1	0	0.5		0	0	0
	C5	1	11.2	22.5		0	0.21964713	0

Table 20. Selfish Negotiation Model results for System 12 (Accepted by SoS)

	Capabilities	Deadline $SoS.d_i$	Funding $SoS.f_i$	Performance $SoS.p_i$		Deadline $Systemj.\Delta d_i$	Funding $Sustemj.\Delta f_i$	Performance $Systemj.\Delta p_i$
The Offer by SoS to Sys 12	C1	1	0	1	System 12 response: Accepted by SoS to provide capabilities C1 and C5	0	0	0
	C2	1	10.35	18.5		0	0.0164735	0
	C3	0	0	0		0	0	0
	C4	0	0	0.5		0	0	0
	C5	1	10.6	21		0	0.14676918	0

Table 21. Selfish Negotiation Model results for System 13 (Accepted by SoS)

	Capabilities	Deadline $SoS.d_i$	Funding $SoS.f_i$	Performance $SoS.p_i$		Deadline $Systemj.\Delta d_i$	Funding $Sustemj.\Delta f_i$	Performance $Systemj.\Delta p_i$
The Offer by SoS to Sys 13	C1	0	0	0	System 13 response: Rejected by SoS to provide capabilities C1 and C5	0	0	0
	C2	1	18	41		2	0.62534576	0
	C3	0	0	0		0	0	0
	C4	0	0	0		0	0	0
	C5	1	18	42		1	0.28538953	0

The system13 can meet the performance requirement on the capability C2 and C5 but requests an extension of deadline for 2nd wave on C2 and an additional funding of 0.6235 units. Since the System is not ready to participate in in the current wave and needs more time (upto the 3rd wave of SoS) the SoS agent rejects the offer of negotiation.

Table 22. Selfish Negotiation Model results for System 14 (Accepted by SoS)

	Capabilities	Deadline $SoS.d_i$	Funding $SoS.f_i$	Performance $SoS.p_i$		Deadline $Systemj.\Delta d_i$	Funding $Sustemj.\Delta f_i$	Performance $Systemj.\Delta p_i$
The Offer by SoS to Sys 14	C1	**1**	**0**	**0.5**	System 14 response: Accepted by SoS to provide capabilities C1 and C5	**0**	**0**	**-0.5**
	C2	1	0	0.5		0	0	-0.5
	C3	**1**	**13**	**12**		**0**	**0**	**0**
	C4	**1**	**0**	**0**		**0**	**0**	**0**
	C5	1	13.45	13.5		0	0	0

Table 23. Selfish Negotiation Model results for System 18 (Accepted by SoS)

	Capabilities	Deadline $SoS.d_i$	Funding $SoS.f_i$	Performance $SoS.p_i$		Deadline $Systemj.\Delta d_i$	Funding $Sustemj.\Delta f_i$	Performance $Systemj.\Delta p_i$
The Offer by SoS to Sys 18	C1	**1**	**0**	**1**	System 18 response: Accepted by SoS to provide capabilities C1 and C5	**0**	**0**	**-1**
	C2	1	0	1		0	0	-1
	C3	**1**	**0**	**0**		**0**	**0**	**0**
	C4	**1**	**2.5**	**15**		**0**	**0**	**-1.95399E-14**
	C5	1	2.75	18		0	0	3.55271E-15

Table 24 Selfish Negotiation Model results for System 21 (Accepted by SoS)

	Capabilities	Deadline $SoS.d_i$	Funding $SoS.f_i$	Performance $SoS.p_i$		Deadline $Systemj.\Delta d_i$	Funding $Sustemj.\Delta f_i$	Performance $Systemj.\Delta p_i$
The Offer by SoS to Sys 21	C1	1	1	2	System 21 response: Accepted by SoS to provide capabilities C1 and C5	0	0	0
	C2	1	0.7	1.5		0	0	0
	C3	0	0	0.5		0	0	0
	C4	1	1	0.5		0	0	0
	C5	1	7.2	19.5		0	0.28627196	0

Table 17 above implies that no changes are required for the deadlines and performance for capability 5. Overall, we also see reduced needs for funding.

Table 25 Selfish Negotiation Model results for System 22 (Accepted by SoS)

	Capabilities	Deadline $SoS.d_i$	Funding $SoS.f_i$	Performance $SoS.p_i$		Deadline $Systemj.\Delta d_i$	Funding $Sustemj.\Delta f_i$	Performance $Systemj.\Delta p_i$
The Offer by SoS to Sys 22	C1	0	0	2	System 21 response: Accepted by SoS to provide capabilities C1 and C5	0	0	-2
	C2	1	0.35	1.5		0	0	-1.5
	C3	1	1	0.5		0	0	-0.5
	C4	0	0	0.5		0	0	-0.5
	C5	1	5.45	19.5		0	0	-1.29958E-11

After the end of the first wave only System 13 is rejected due to the reasons give above. All other systems are selected and Since System 13 did not have any interface with the other participating systems, the interface matrix does not get affected as well.

Final Architecture is shown in Figure 44 below:

S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{14}	S_{15}	S_{16}	S_{17}	S_{18}	S_{19}	S_{20}	S_{21}	S_{22}
1	1	0	0	0	0	1	1	0	0	1	1	1	0	0	0	1	0	0	1	1

Figure 44. Final Architecture

The interface architecture remains the same as before.

5.3 Opportunistic Model Results

In the results below η is set to 0.2, and both *l* =1.9 and *m* =1.1. This implies that the system assumes a high risk and estimating more money needed to provide the capabilities. One of the behavior factors "l" is kept high to make system very slow in operation while the other "m" is kept low for faster output. Although , in the negotiation process, the values of η, l and m can be changed at will to obtain a large number of systems, and thereby the SoS has the ability to select a suitable system that is sufficiently selfless and fast in its actions. This selection of values for the parameters is chosen to have a tradeoff between all three. The results could be interpreted as follows:

Table 26 contains information regarding a bilateral negotiation between SoS and System 1 through an opportunistic negotiation model. System 1 can provide Capability 1 (C1) and/or (C2) but no more. The values in the first row corresponding to C1 indicate that SoS requires System 1 to join in the first wave of the SoS architecture. SoS is providing a funding of 10.2 units to acquire C1 and demands performance level of 13.5 units for the same. The response to this offer of SoS can be read in the adjacent columns consisting of delta values of attributes. The response is formulated in terms of delta change.

Table 26 Opportunistic Negotiation Model results for System 1 (Accepted by SoS)

	Capabilities	**Deadline** $SoS.d_i$	**Funding** $SoS.f_i$	**Performance** $SoS.p_i$		**Deadline** $Systemj.\Delta d_i$	**Funding** $Sustemj.\Delta f_i$	**Performance** $Systemj.\Delta p_i$
The Offer by SoS to Sys 1	**C1**	1	10.2	13.5	**System 1 response: Accepted by SoS to provide capabilities C1 and C5**	0	-1.23845324	-2.372174142
	C2	0	0	0		0	0	0
	C3	0	0	0		0	0	0
	C4	0	0	0		0	0	0
	C5	1	10.55	16		0	1.280949185	-2.81146565

The delta values correspond to ($delta = System's\ response - SoS\ offer$). Hence a positive value of "1"in deadline means that System is not ready to participate in the first wave but will be ready by the second wave. Equivalently the funding delta value is negative 1.238 meaning the System 1 has offered to work for a lesser amount from SoS for providing C1. The Performance delta value is -2.37 which can be interpreted as the System 1 offers lesser performance than required but at a lower cost.

Similarly for C5, SoS asks the System 1 to join in the first wave, has provision of 10.55 units of funding to extract a performance of 16 units. Based on the delta values of each attributes and the SoS negotiation thresholds, System 1 is accepted to be a part of the first wave of the architecting process. Table 27 through Table 36 present the results for the rest of the systems.

Table 27 Opportunistic Negotiation Model results for System 2 (Accepted by SoS)

	Capabilities	Deadline $SoS.d_i$	Funding $SoS.f_i$	Performance $SoS.p_i$		Deadline $Systemj.\Delta d_i$	Funding $Sustemj.\Delta f_i$	Performance $Systemj.\Delta p_i$
The Offer by SoS to Sys 2	**C1**	**1**	**10.1**	**11**	**System 2 response: Accepted by SoS to provide capabilities C1 and C5**	**0**	**-0.8702365**	**-2.010443534**
	C2	1	0	0		0	0	0
	C3	**0**	**0**	**0**		**0**	**0**	**0**
	C4	**1**	**0**	**0**		**0**	**0**	**0**
	C5	1	10.2	12		0	-0.8788527	-2.193211127

Table 28 Opportunistic Negotiation Model results for System 7 (Accepted by SoS)

	Capabilities	Deadline $SoS.d_i$	Funding $SoS.f_i$	Performance $SoS.p_i$		Deadline $Systemj.\Delta d_i$	Funding $Sustemj.\Delta f_i$	Performance $Systemj.\Delta p_i$
The Offer by SoS to Sys 7	**C1**	**1**	**2.2**	**13.5**	**System 7 response: Accepted by SoS to provide capabilities C1 and C5**	**0**	**-0.1895565**	**-2.467362518**
	C2	1	0	1		0	0	0
	C3	**1**	**0**	**0**		**0**	**0**	**0**
	C4	**1**	**0**	**0.5**		**0**	**0**	**0**
	C5	1	2.45	16		0	-0.211097	-2.924281503

Table 29 Opportunistic Negotiation Model results for System 8 (Accepted by SoS)

	Capabilities	Deadline $SoS.d_i$	Funding $SoS.f_i$	Performance $SoS.p_i$		Deadline $Systemj.\Delta d_i$	Funding $Sustemj.\Delta f_i$	Performance $Systemj.\Delta p_i$
The Offer by SoS to Sys 8	C1	1	15	5	System 8 response: Accepted by SoS to provide capabilities C1 and C5	0	-1.8212548	-0.878583016
	C2	1	0	0.5		0	0	0
	C3	0	0	0.5		0	0	0
	C4	1	0	0		0	0	0
	C5	1	0.25	2		0	0	0

Table 30. Opportunistic Negotiation Model results for System 11 (Accepted by SoS)

	Capabilities	Deadline $SoS.d_i$	Funding $SoS.f_i$	Performance $SoS.p_i$		Deadline $Systemj.\Delta d_i$	Funding $Sustemj.\Delta f_i$	Performance $Systemj.\Delta p_i$
The Offer by SoS to Sys 11	C1	1	0	1.5	System 11 response: Accepted by SoS to provide capabilities C1 and C5	0	0	0
	C2	1	10.7	19.5		0	0.44188368	-4.061060409
	C3	1	0	0.5		0	0	0
	C4	1	0	0.5		0	0	0
	C5	1	11.2	22.5		0	0.46253246	-4.685838933

Table 31. Opportunistic Negotiation Model results for System 12 (Accepted by SoS)

	Capabilities	Deadline $SoS.d_i$	Funding $SoS.f_i$	Performance $SoS.p_i$		Deadline $Systemj.\Delta d_i$	Funding $Sustemj.\Delta f_i$	Performance $Systemj.\Delta p_i$
The Offer by SoS to Sys 12	C1	1	0	1	System 12 response: Accepted by SoS to provide capabilities C1 and C5	0	0	0
	C2	1	10.35	18.5		0	-1.9020002	-3.020057917
	C3	0	0	0		0	0	0
	C4	0	0	0.5		0	0	0
	C5	1	10.6	21		0	-1.9479422	-3.428173851

Table 32. Opportunistic Negotiation Model results for System 13 (Accepted by SoS)

	Capabilities	Deadline $SoS.d_i$	Funding $SoS.f_i$	Performance $SoS.p_i$		Deadline $Systemj.\Delta d_i$	Funding $Sustemj.\Delta f_i$	Performance $Systemj.\Delta p_i$
The Offer by SoS to Sys 13	C1	0	0	0	System 13 response: Rejected by SoS to provide capabilities C1 and C5	0	0	0
	C2	1	18	41		0	-4.2711864	-6.254237288
	C3	0	0	0		0	0	0
	C4	0	0	0		0	0	0
	C5	1	18	42		0	-4.2711864	-6.406779661

Table 33 Opportunistic Negotiation Model results for System 14 (Accepted by SoS)

	Capabilities	Deadline $SoS.d_i$	Funding $SoS.f_i$	Performance $SoS.p_i$		Deadline $Systemj.\Delta d_i$	Funding $Sustemj.\Delta f_i$	Performance $Systemj.\Delta p_i$
The Offer by SoS to Sys 14	C1	**1**	**0**	**0.5**	System 14 response: Accepted by SoS to provide capabilities C1 and C5	**0**	**0**	**0**
	C2	1	0	0.5		0	0	0
	C3	**1**	**13**	**12**		**0**	**2.81719128**	**-2.920096852**
	C4	**1**	**0**	**0**		**0**	**0**	**0**
	C5	1	13.45	13.5		0	2.914709444	-3.285108959

Table 34 Opportunistic Negotiation Model results for System 18 (Accepted by SoS)

	Capabilities	Deadline $SoS.d_i$	Funding $SoS.f_i$	Performance $SoS.p_i$		Deadline $Systemj.\Delta d_i$	Funding $Sustemj.\Delta f_i$	Performance $Systemj.\Delta p_i$
The Offer by SoS to Sys 18	C1	**1**	**0**	**1**	System 18 response: Accepted by SoS to provide capabilities C1 and C5	**0**	**0**	**0**
	C2	1	0	1		0	0	0
	C3	**1**	**0**	**0**		**0**	**0**	**0**
	C4	**1**	**2.5**	**15**		**0**	**-0.11913**	**-2.857044007**
	C5	1	2.75	18		0	-0.131043	-3.428452808

Table 35 Opportunistic Negotiation Model results for System 21 (Accepted by SoS)

	Capabilities	Deadline $SoS.d_i$	Funding $SoS.f_i$	Performance $SoS.p_i$		Deadline $Systemj.\Delta d_i$	Funding $Sustemj.\Delta f_i$	Performance $Systemj.\Delta p_i$
The Offer by SoS to Sys 21	C1	**1**	**1**	**2**	System 21 response: Accepted by SoS to provide capabilities C1 and C5	**0**	**0**	**0**
	C2	1	0.7	1.5		0	0	0
	C3	**0**	**0**	**0.5**		**0**	**0**	**0**
	C4	**1**	**1**	**0.5**		**0**	**0**	**0**
	C5	1	7.2	19.5		0	0.29734229	-4.061060409

Table 36 Opportunistic Negotiation Model results for System 22 (Accepted by SoS)

	Capabilities	Deadline $SoS.d_i$	Funding $SoS.f_i$	Performance $SoS.p_i$		Deadline $Systemj.\Delta d_i$	Funding $Sustemj.\Delta f_i$	Performance $Systemj.\Delta p_i$
The Offer by SoS to Sys 22	**C1**	**0**	**0**	**2**	**System 21 response: Accepted by SoS to provide capabilities C1 and C5**	**0**	**0**	**0**
	C2	1	0.35	1.5		0	0	0
	C3	**1**	**1**	**0.5**		**0**	**0**	**0**
	C4	**0**	**0**	**0.5**		**0**	**0**	**0**
	C5	1	5.45	19.5		0	-0.6617226	-3.426473761

After the end of the first wave all systems are selected and all interfaces are retained as well. So the interface matrix does not get affected as well.

Final Architecture is shown in Figure 45 below:

S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	S_{16}	S_{17}	S_{18}	S_{19}	S_{20}	S_{21}	S_{22}
1	1	0	0	0	0	1	1	0	0	1	1	1	1	0	0	0	1	0	0	1	1

Figure 45. Final Architecture

The interface architecture remains the same as before.

5.4 Cooperative Model Results

In the results below, cooperativeness of the system denoted in the model by "V" is set equal to 0.8. This implies that the system assumes a high cooperativeness. The innate cooperative behavior is assumed to be 3 times higher or more than the constrained behavior. Hence, the weight for innate cooperative behavior is 0.7 and 03 for constrained behavior of the individual systems. Both the cooperativeness "V" and the weightage can have different values for each individual system. Although to present results in a simple fashion, they are kept the same throughout the calculations. The results are shown in Table 37 thought Table 47:

Table 37 Cooperative Negotiation Model results for System 1 (Accepted by SoS)

	Capabilities	Deadline $SoS.d_i$	Funding $SoS.f_i$	Performance $SoS.p_i$		Deadline $Systemj.\Delta d_i$	Funding $Sustemj.\Delta f_i$	Performance $Systemj.\Delta p_i$
The Offer by SoS to Sys 1	C1	1	10.2	13.5	System 1 response: Accepted by SoS to provide capabilities C1 and C5	**0**	**-1.428**	**-0.937058824**
	C2	0	0	0		**0**	0	0
	C3	0	0	0		**0**	**0**	**0**
	C4	0	0	0		**0**	**0**	**0**
	C5	1	10.55	16		**1**	3.67892463	6.708823529

Table 38 Cooperative Negotiation Model results for System 2 (Accepted by SoS)

	Capabilities	Deadline $SoS.d_i$	Funding $SoS.f_i$	Performance $SoS.p_i$		Deadline $Systemj.\Delta d_i$	Funding $Sustemj.\Delta f_i$	Performance $Systemj.\Delta p_i$
The Offer by SoS to Sys 2	C1	**1**	**10.1**	**11**	System 2 response: Accepted by SoS to provide capabilities C1 and C5	**0**	**-1.414**	**-0.50875**
	C2	1	0	0		0	0	0
	C3	**0**	**0**	**0**		**0**	**0**	**0**
	C4	**1**	**0**	**0**		**0**	**0**	**0**
	C5	1	10.2	12		1	3.85984158	5.665990099

Table 39 Cooperative Negotiation Model results for System 7 (Accepted by SoS)

	Capabilities	Deadline $SoS.d_i$	Funding $SoS.f_i$	Performance $SoS.p_i$		Deadline $Systemj.\Delta d_i$	Funding $Sustemj.\Delta f_i$	Performance $Systemj.\Delta p_i$
The Offer by SoS to Sys 7	C1	1	2.2	13.5	System 7 response: Accepted by SoS to provide capabilities C1 and C5	0	-0.308	-0.624375
	C2	1	0	1		0	0	0
	C3	1	0	0		0	0	0
	C4	1	0	0.5		0	0	0
	C5	1	2.45	16		1	0.9461733	7.679090909

Table 40. Cooperative Negotiation Model results for System 8 (Accepted by SoS)

	Capabilities	Deadline $SoS.d_i$	Funding $SoS.f_i$	Performance $SoS.p_i$		Deadline $Systemj.\Delta d_i$	Funding $Sustemj.\Delta f_i$	Performance $Systemj.\Delta p_i$
The Offer by SoS to Sys 8	C1	1	15	5	System 8 response: Accepted by SoS to provide capabilities C1 and C5	0	-2.1	-0.347058824
	C2	1	0	0.5		0	0	0
	C3	0	0	0.5		0	0	0
	C4	1	0	0		0	0	0
	C5	1	0.25	2		0	0	0

Table 41. Cooperative Negotiation Model results for System 11 (Accepted by SoS)

	Capabilities	Deadline $SoS.d_i$	Funding $SoS.f_i$	Performance $SoS.p_i$		Deadline $Systemj.\Delta d_i$	Funding $Sustemj.\Delta f_i$	Performance $Systemj.\Delta p_i$
The Offer by SoS to Sys 11	C1	1	0	1.5	System 11 response: Accepted by SoS to provide capabilities C1 and C5	0	0	0
	C2	1	10.7	19.5		0	-1.498	0.195
	C3	1	0	0.5		0	0	0
	C4	1	0	0.5		0	0	0
	C5	1	11.2	22.5		1	4.12173956	11.65528037

Table 42 . Cooperative Negotiation Model results for System 12 (Accepted by SoS)

	Capabilities	Deadline $SoS.d_i$	Funding $SoS.f_i$	Performance $SoS.p_i$		Deadline $Systemj.\Delta d_i$	Funding $Sustemj.\Delta f_i$	Performance $Systemj.\Delta p_i$
The Offer by SoS to Sys 12	C1	1	0	1	System 12 response: Accepted by SoS to provide capabilities C1 and C5	0	0	0
	C2	1	10.35	18.5		0	-1.449	-2.005789474
	C3	0	0	0		0	0	0
	C4	0	0	0.5		0	0	0
	C5	1	10.6	21		1	3.87164573	8.333399441

Table 43 Cooperative Negotiation Model results for System 13 (Accepted by SoS)

	Capabilities	Deadline $SoS.d_i$	Funding $SoS.f_i$	Performance $SoS.p_i$		Deadline $Systemj.\Delta d_i$	Funding $Sustemj.\Delta f_i$	Performance $Systemj.\Delta p_i$
The Offer by SoS to Sys 13	C1	0	0	0	System 13 response: Rejected by SoS to provide capabilities C1 and C5	0	0	0
	C2	1	18	41		0	0	0
	C3	0	0	0		0	0	0
	C4	0	0	0		0	0	0
	C5	1	18	42		0	0	0

Table 44 Cooperative Negotiation Model results for System 14 (Accepted by SoS)

	Capabilities	Deadline $SoS.d_i$	Funding $SoS.f_i$	Performance $SoS.p_i$		Deadline 0	Funding 0	Performance 0
The Offer by SoS to Sys 14	C1	1	0	0.5	System 14 response: Accepted by SoS to provide capabilities C1 and C5	0	0	0
	C2	1	0	0.5		0	-1.82	2.28
	C3	1	13	12		0	0	0
	C4	1	0	0		1	5.04386496	9.517615385
	C5	1	13.45	13.5		0	0	0

Table 45 Cooperative Negotiation Model results for System 18 (Accepted by SoS)

	Capabilities	Deadline $SoS.d_i$	Funding $SoS.f_i$	Performance $SoS.p_i$		Deadline $Systemj.\Delta d_i$	Funding $Sustemj.\Delta f_i$	Performance $Systemj.\Delta p_i$
The Offer by SoS to Sys 18	C1	1	0	1	System 18 response: Accepted by SoS to provide capabilities C1 and C5	0	0	0
	C2	1	0	1		0	0	0
	C3	1	0	0		0	0	0
	C4	1	2.5	15		0	-0.35	-0.3
	C5	1	2.75	18		1	1.02666667	8.88

Table 46 Cooperative Negotiation Model results for System 21 (Accepted by SoS)

	Capabilities	Deadline $SoS.d_i$	Funding $SoS.f_i$	Performance $SoS.p_i$		Deadline $Systemj.\Delta d_i$	Funding $Sustemj.\Delta f_i$	Performance $Systemj.\Delta p_i$
The Offer by SoS to Sys 21	C1	1	1	2	System 21 response: Accepted by SoS to provide capabilities C1 and C5	0	0	0
	C2	1	0.7	1.5		0	0	0
	C3	0	0	0.5		0	0	0
	C4	1	1	0.5		0	0	0
	C5	1	7.2	19.5		0	-1.008	0.195

Table 47 Cooperative Negotiation Model results for System 22 (Accepted by SoS)

	Capabilities	Deadline $SoS.d_i$	Funding $SoS.f_i$	Performance $SoS.p_i$		Deadline $Systemj.\Delta d_i$	Funding $Sustemj.\Delta f_i$	Performance $Systemj.\Delta p_i$
The Offer by SoS to Sys 22	C1	0	0	2	System 21 response: Accepted by SoS to provide capabilities C1 and C5	0	0	0
	C2	1	0.35	1.5		0	0	0
	C3	1	1	0.5		0	0	0
	C4	0	0	0.5		0	0	0
	C5	1	5.45	19.5		0	-0.763	-1.353529412

After the end of the first wave all systems are selected and all interfaces are retained such that the interface matrix does not get affected.

The final architecture is shown in Figure 46 below:

S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	S_{16}	S_{17}	S_{18}	S_{19}	S_{20}	S_{21}	S_{22}
1	1	0	0	0	0	1	1	0	0	1	1	1	1	0	0	0	1	0	0	1	1

Figure 46. Final Architecture

The interface architecture remains the same as before.

These results drive the conclusion that all system via a cooperative negotiation model are selected to participate in the SoS. This concludes the first wave as well.

	Arch Quality
Architecture	**3.6419**
Performance	2.592249914
Affordability	3.577980575
Flexibility	3
Robustness	3.757302522

6 Concluding Remarks and Future Work

The current analysis environment has matured to the point where it could support SoS analysis and decision-making, which would identify new opportunities to improve the SoS analysis tools. The next step is to create the demonstration and presentation materials necessary to describe the capabilities of the ABM, and provide an overview of analysis tools to potential users identified by the sponsor. In the next phase of the project as a new research task, team shall create the demonstration and presentation materials necessary to describe the capabilities of the toolset, and provide an overview of analysis tools to potential users to be identified by the sponsor.

In addition, there are several areas of analysis, which we would like to continue our previous investigations, as well as explore new areas such as:

- What is the impact of different constituent system perspectives regarding participating in the SoS on the overall mission effectiveness of the SoS?

- How do differing levels of cooperativeness in participating in the SoS impact the ability and timeliness of a group to agree on a SoS or system architecture? Or impact the ability to effectively use the architecture already in place?

- How should decision-makers incentivize systems to participate in SoS, and better understand the impact of these incentives during SoS development and effectiveness?

Missouri S&T team is planning to make any necessary changes to apply and validate the utility of the tools to address these analysis requirements. In addition to providing transition support for customer organizations, Missouri S&T team like to plan on investigating the potential to incorporate the Missouri S&T SoS analysis tools in the analytic workbench being developed by Purdue University. Although the

Purdue and Missouri S&T tools may end up serving different analysis communities, we believe there could be significant benefits for managing the support of the tools in similar ways.

7 Bibliography

[1] D. ASD(NII), "DoD Architecture Framework Version 2.0 (DoDAF V2.0)," Department of Defense, Washington DC, 2009.

[2] J. P. Dauby and S. Upholzer, "Exploring Behavioral Dynamics in Systems of Systems," in *Complex Adaptive Systems, Procedia Computer Science, vol 6*, Elsevier, 2011, pp. 34-39.

[3] P. Acheson, "Methodology for Object Oriented System Architecture Development," in *IEEE Systems Conference*, 2010.

[4] E. Bonabeau, "Agent based Modeling: Methods and Techniques for Simulating Human Systems,," *Proceedings of the National Academy of Sciences, Vol. 99,* pp. 7280-7287, 2002.

[5] N. Kilicay-Ergin, D. Enke and C. Dagli, "Biased trader model and analysis of financial market dynamics," *International Journal of Knowledge-based Intelligent Engineering Systems, Vol. 16,* pp. 99-116, 2012.

[6] W. Weiss, "Dynamic Security: An Agent-based Model for Airport Defense," in *Proceedings of the Winter Simulation Conference*, 2008.

[7] D. Dudenhoeffer and M. Jones, "A Formation Behavior for Large Scale Micro-Robot Force Deployment," in *Proceedings of the 32nd Conference on Winter Simulation*, 2000.

[8] Director Systems and Software Engineering, OUSD (AT&L), "Systems Engineering Guide for Systems of Systems," available from http://www.acq.osd.mil/se/docs/SE-Guide-for-SoS.pdf, 2008.

[9] J. Dahmann, G. Rebovich, J. A. Lane, R. Lowry and K. Baldwin, "An Implementers' View of Systems Engineering for Systems of Systems," in *Proceedings of IEEE International Systems Conference*, Montreal, 2011.

[10] J. A. Lane and T. Bohn, "Using SysML Modeling to Understand and Evolve System of Systems," *Systems Engineering,* vol. 16, no. 1, pp. 87-98, 2012.

[11] Y. Y. Haimes, "Modeling Complex Systems of Systems with Phantom System Models," *Systems Engineering,* vol. 15, no. 3, pp. 333-346, 2012.

[12] R. K. Garrett, S. Anderson, N. T. Baron and J. D. Moreland, "Managing the Interstitials, a System of Systems Framework Suited for the Ballistic Missile Defense System," *Systems Engineering,* vol. 14, no. 1, pp. 87-109, 2009.

[13] B. Ge, K. W. Hipel, K. Wang and Y. Chen, "A Data-centric Capability-Focused Approach for System-of-Systems Architecture Modeling and Analysis," *Systems Engineering,* vol. 16, no. 3, pp. 363-377, 2013.

[14] J. P. Dauby and C. H. Dagli, "The canonical decomposition fuzzy comparative methodology for assessing architectures," *IEEE Systems Journal, vol. 5, no. 2,* pp. 244-255, 2011.

[15] S. A. Henson, M. J. Henshaw, V. Barot, C. E. Siemieniuch, H. Dogan, S. L. Lim, C. Ncube, M. Jamshidi and D. DeLaurentis, "Towards a Systems of Systems Engineering EU Strategic Research Agenda," in *IEEE International Conference on System of Systems Engineering*, Maui, 2013.

[16] A. Arnold, B. Boyer and A. Legay, "Contracts and Behavioral Patterns for System of Systems: The EU IP DANSE Approach," 2012.

[17] J. W. Coleman, A. K. Malmos, P. g. Larsen, J. Peleska, R. Hains, Z. Andrews, R. Payne, S. Foster, A. Miyazawa, C. Bertolini and A. Didier, "COMPASS Tool Vision for a System of Systems Collaborative Development Environment," in *Proceedings of the 7th International Conference on System of System Engineering, IEEE SoSE 2012*, 2012.

[18] J. K. S. B. J. P. C. C. M. J. G. J. K. R. W. a. W. W. Bergey, "US Army Workshop on Exploring Enterprise, System of Systems, System, and Software Architectures," 2009.

[19] CJCSI 6212.01F, "Net Ready Key Performance Parameter (NR KPP)," US Dept of Defense, Washington DC, 12 Mar 2012.

[20] A. Singh and C. H. Dagli, "Multi-objective Stochastic Heuristic Method for Trade Space Exploration of a Network Centric System of Systems," in *3rd Annual IEEE International Systems Conference, 2009*, Vancouver, Canada, 2009.

[21] J. Dahmann, K. J. Baldwin and G. Rebovich, "Systems of Systems and Net-Centric Enterprise Systems," in *7th Annual Conference on Systems Engineering Research*, Loughborough, 2009.

[22] J.-H. Ahn, "An Archietcture Description method for Acknowledged System of Systems based on Federated Architeture," in *Advanced Science and Technology Letters 05*, 2012.

[23] S. Y. Han and D. DeLaurentis, "Development Interdependency Modeling for System-of-Systems (SoS) using Bayesian Networks: SoS Management Strategy Planning," in *Procedia Computer Science, Volume 16, 698–707*, Atlanta, 2013.

[24] Trans-Atlantic Research and Education Agenda in Systems of Systems (T-AREA-SOS) Project, "The Systems of Systems Engineering Strategic Research Agenda," Loughborough University, Loughborough, 2013.

[25] R. Pitsko and D. Verma, "Principles for Architecting Adaptable Command and Control," in *New Challenges in Systems Engineering and Architecting*, St. Louis, 2012.

[26] M. W. Schreiner and J. R. Wirthlin, "Challenges using modeling and simulation in architecture," *Procedia Computer Science,* vol. 8, pp. 153-158, 2012.

[27] D. Flanagan and P. Brouse, "System of Systems Requirements Capacity Allocation," *Procedia Computer Science,* vol. 8, pp. 112-117, 2012.

[28] R. E. Giachetti, "A Flexible Approach to Realize an Enterprise Architecture," *Procedia Computer Science,* vol. 8, pp. 147-152, 2012.

[29] J. Clune, J.-B. Mouret and H. Lipson, "The evolutionary origins of modularity," *Proceedings of the Royal Society - B,* vol. 280, p. 2863, 2013.

[30] J. A. Christian III, "A Quantitative Approach to Assessing System Evolvability," NASA Johnson Space Center, Houston, 2004.

[31] M. J. Kinnunen, "Complexity Measures for System Architecture Models," MIT: System Design and Management Program Masters Thesis, Cambridge, 2006.

[32] Y. Mordecai and D. Dori, "I5: A Model-Based Framework for Architecting System-of-Systems Interoperability, Interconnectivity, Interfacing, Integration, and Interaction," in *International Symposium of the International Council on Systems Engineering (INCOSE)*, Philadelphia, 2013.

[33] D. N. Fry and D. A. DeLaurentis, "Measuring Net-Centricity," in *Proceedings of the 6th International Conference on System of Systems Engineering*, Albuquerque, 2011.

[34] N. Ricci, A. M. Ross, D. H. Rhodes and M. E. Fitzgerald, "Considering Alternative Strategies for Value Sustainment in Systems-of-Systems (Draft)," Systems Engineering Advancement Research Initiative, Cambridge MA, 2013.

[35] P. Acheson, L. Pape, C. Dagli, N. Kilcay-Ergin, J. Columbi and K. Haris, "Understanding System of Systems Development Using an Agent-Based Wave Model," in *Complex Adaptive Systems, Publication 2*, Washington D.C., 2012.

[36] J. M. Lafleur, "A Markovian State-SpaceFramework for Integrating Flexibility into Space System Design Decisions," Georgia Institute of Technology School of Aerospace Engineering Doctoral Thesis, Atlanta, 2012.

[37] K. Deb and H. Gupta, "Introducing Robustness in Multi-Objective Optimization," *Evolutionary Computation,* vol. 14, no. 4, pp. 463-494, 2006.

[38] Y. Singer, "Dynamic Measure of Network Robustness," in *IEEE 24th Conference of Electrical and Electronic Engineers in Israel*, 2006.

[39] C. Smartt and S. Ferreira, "Constructing a General Framework for Systems Engineering Strategy," *Systems Engineering,* vol. 15, no. 2, pp. 140-152, 2012.

[40] O.-Y. Yu, S. D. Gulkema, J.-L. Briaud and D. Burnett, "Sensitivity Analysis for Multi-Attribute System Selection Problems in Onshore Environmentally Friendly Drilling (EFD)," *Systems Engineering,* vol. 15, no. 2, pp. 153-171, 2011.

[41] S. Jackson and T. L. J. Ferris, "Resilience Principles for Engineered Systems," *Systems Engineering,* vol. 16, no. 2, pp. 152-164, 2013.

[42] J. M. Mendel, "On KM Algroithms for Solving Type-2 Fuzzy Set Problems," *IEEE Transactions on Fuzzy Systems,* vol. 21, no. 3, pp. 426-446, 2013.

[43] C. Li and T.-W. Chiang, "Complex Neurofuzzy ARIMA Forecasting - A New Approach Using Complex Fuzzy Sets," *IEEE Transactions on Fuzzy Systems,* vol. 21, no. 3, pp. 567-584, 2013.

[44] S. Kraus, "Automated negotiation and decision making in multiagent environments," *Easss 2086,* pp. 150-172, 2001.

[45] T. Wanyama and B. H. Far, "A Protocol for Multi-agent Negotiation in a Group choice Decision Making Process," *Journal of Network and Computer Applications,* vol. 30, pp. 1173-1195, 2007.

[46] S. Parsons and M. Wooldridge, "Languages for Negotiation," in *Proceedings of the Fourteenth European Conference on Artificial Intelligence (ECAI-2000)*, 2000.

[47] S. S. Fatima, M. Wooldridge and N. Jennings, "An Agenda-based Framework for Multi-issue negotiation," *Artificial Intelligence,* vol. 152, pp. 1-45, 2004.

[48] C. M. Jonker, V. Robu and J. Treur, "An Agent Architecture for Multi-attribute Negotiation Using Incomplete Preference Information," *Auton Agent Multi-agent System,* vol. 15, pp. 221-252, 2007.

[49] N. K. C. Krothapali and A. V. Deshmukh, "Design of Negotiation Protocols for Multi-agent Manufacturing Systems," *International Journal of Production Research,* vol. 37, no. 7, pp. 1601-1624, 1999.

[50] P. Acheson, C. Dagli and N. Kilicay-Ergin, "Fuzzy Decision Analysis in Negotiation between the System of Systems Agent and the System Agent in an Agent Based Model," *International Journal of Soft Computing and Software Engineering [JSCSE],* March 2013.

[51] S. Kraus, "An Overview of Incentive Contracting," *Artificial Intelligence,* vol. 83, pp. 297-346, 1996.

[52] G. R. Djavanshir, A. Alavizadeh and M. J. Tarokh, "From System-of-Systems to Meta-Systems: Ambiguities and Challenges," in *System of Systems*, 2012.

[53] N. Kilicay Ergin and C. H. Dagli, in *System of Systems: Innovations for the 21st Century*, Wiley & Sons, Inc., 2008.

[54] G. Anandalingam and T. L. Friesz, "Hierarchical optimization: An introduction," *Annals of Operations Research,* vol. 34, no. 1, pp. 1-11, 1992.

[55] A. Migdalas, P. M. Pardalos and P. Värbrand, Multilevel Optimization: Algorithms and Applications, Dordrecht: Kluwer Academic Publishers, 1998.

[56] P. Hansen , B. Jaumard and G. Savard, "New branch-and-bound rules for linear bilevel programming," *SIAM Journal on Scientific and Statistical Computing,* vol. 13, no. 5, p. 1194–1217, 1992.

[57] K. Deb and A. Sinha, "Solving bilevel multi-objective optimization problems using evolutionary algorithms," in *Evolutionary Multi-Criterion Optimization. Lecture Notes in Computer Science*, Berlin, Springer, 2009, p. 110–124.

[58] M. J. Alves, S. Dempe and J. J. Júdice, "Computing the pareto frontier of a bi-objective bi-level linear problem using a multiobjective mixed-integer programming algorithm," *Optimization,* vol. 61, no. 3, p. 335–358, 2012.

[59] G. Eichfelder, "Multiobjective bilevel optimization," *Mathematical Programming,* vol. 123, no. 2, pp. 419-449, 2010.

[60] M. Li, D. Lin and S. Wang, "Solving a type of biobjective bilevel programming problem using NSGA-II," *Computers and Mathematics with Applications,* vol. 59, no. 2, pp. 706-715, 2010.

[61] L. Jia, Y. Wang and L. Fan, "Uniform Design Based Hybrid Genetic Algorithm for Multiobjective Bilevel Convex Programming," in *Proceedings of the Seventh International Conference on Computational Intelligence and Security*, 2011.

[62] D. Konur and . M. M. Golias, "Cost-stable truck scheduling at a cross-dock facility with unknown truck arrivals: A meta-heuristic approach," *Transportation Research Part E,* vol. 49, no. 1, pp. 71-91, 2013.

[63] T. Zhang, T. Hu, Y. Zheng and X. Guo, "An Improved Particle Swarm Optimiza-tion for Solving Bilevel Multiobjective Programming Problem," *Journal of Applied Mathematics,,* vol. 2012, 2012.

[64] T. Zhang, T. Hu, C. Jia-wei, Z. wang and X. Guo, "Solving Bilevel Multiobjective Programming Problem by Elite Quantum Behaved Particle Swarm Optimization," *Journal of Applied Mathematics,* vol. 2012, 2012.

[65] M. S. Osman, M. A. Abo-Sinna, A. H. Amer and O. E. Emam, "A multi-level non-linear multi-objective decision-making under fuzziness," *Applied Mathematics and Computation,* vol. 153, no. 1, pp. 239-252, 2004.

[66] J. Lu, G. Zhang and T. Dillon, " Fuzzy multi-objective bilevel decision making by an approximation kth-best approach," *Multiple-Valued Logic and Soft Computing,* vol. 14, p. 205–232, 2008.

[67] G. Zhang and J. Lu, "Fuzzy bilevel programming with multiple objectives and cooperative multiple followers," *Global Optimization,* vol. 47, pp. 403-419, 2010.

[68] Y. Gao, G. Zhang and J. Lu, "A fuzzy multi-objective bilevel decision support system," *International Journal of Information Technology and Decision Making,* vol. 8, no. 1, p. 93–108, 2009.

[69] I. A. Baky, "Fuzzy goal programming algorithm for solving decentralized bi-level multi-objective programming problems," *Fuzzy Sets and Systems,* vol. 160, p. 2701–2713, 2009.

[70] I. A. Baky, "Solving multi-level multi-objective linear programming problems through fuzzy goal programming approach," *Applied Mathematical Modelling,* vol. 34, p. 2377–2387, 2010.

[71] Y. Zheng, Z. Wan and G. Wang, "A fuzzy interactive method for a class of bilevel multiobjective programming problem," *Expert Systems with Applications,* vol. 38, p. 10384–10388, 2011.

[72] C. O. Pieume , P. Marcotte , L. P. Fotso and P. Siarry. , "Solving bilevel linear multiobjective programming problems," *American Journal of Operations Research,* vol. 1, p. 214–219, 2011.

[73] G. Zhan , J. Lu and G. Ya, "An algorithm for fuzzy multi-objective multi-follower partial cooperative bilevel programming," *Journal of Intelligent & Fuzzy Systems,* vol. 19, p. 303–319, 2008.

[74] G. Zhan, J. Lu and G. Ya, "Fuzzy bilevel programming: multi-objective and multi-follower with shared variables," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems,* vol. 16, pp. 105-133, 2008.

[75] X. Shi and H. S. Xia, "Model and Interactive Algorithm of Bi-Level Multi-Objective Decision-Making with Multiple Interconnected Decision Makers," *Journal of Multi-criteria Decision Analysis,* vol. 10, pp. 27-34, 2001.

[76] M. A. Abo-Sinna and I. A. Baky, "Interactive balance space approach for solving multi-level multi-objective programming problems," *Information Sciences,* vol. 177, p. 3397–3410, 2007.

[77] A. B. Cara, C. Wagner, H. Hagras, H. Pomares and I. Rojas, "Multiobjective Optimization and comparison of Nonsingleton Type-1 and Singleton Interval Type-2 Fuzzy Logic Systems," *IEEE Transactions on Fuzzy Systems,* vol. 21, no. 3, pp. 459-476, 2013.

[78] J.-Q. Wang and H.-Y. Zhang, "Multicriteria Decision-Making Approach Based on Atanassov's Intuitionistic Fuzzy Sets With Incomplete Certain Information on Weights," *IEEE Transactions on Fuzzy Systems,* vol. 21, no. 3, pp. 510-515, 2013.

[79] J. A. Sanz, A. Fernandez, H. Bustince and F. Herrara, "IVTURS: A Linguistic Fuzzy Rule-Based Classification System Based On a New Interval-Valued Fuzzy Reasoning Method with Tuniing and Rule Selection," *IEEE Transactions on Fuzzy Systems,* vol. 21, no. 3, pp. 399-411, 2013.

[80] D. Fogel, Evolutionary Computation, New Jersey: John Wiley and Sons, 2006.

[81] S. Sumathi and P. Surekha, Computational Intelligence Paradigms: Theory & Applications Using MATLAB, Boca Raton FL: CRC Press, 2010.

[82] W. Pedrycz, P. Ekel and R. Parreiras, Fuzzy Multicriteria Decision Making; Models, Methods and Applications, West Sussex: John Wiley & Sons, 2011.

[83] A. Alfaris, *The Evolutionary Design Model (EDM) for the Design of Complex Engineered Systems: Masdar City as a Case Study,* Massachusetts Institute of Technology, 2009.

[84] P. J. Fleming and C. M. Fonseca, "An overview of evolutionary algorithms in multiobjective optimization," *Evolutionary Computation,* vol. 3, no. 2, pp. 1-16, 1995.

[85] C. A. Coello and G. B. Lamont, "An Introduction to Multi-Objective Evolutionar Algorithms and Their Applications," *Applications of Multi-Objective Evolutionary Algorithms,* vol. 1, pp. 1-28., 2004.

[86] K. Deb, "Multi-objective Optimization using Evolutionary Algorithms," vol. 16, no. Wiley, 2001.

[87] M. Gries, "Methods for Evaluating and Covering the Design Space during Early Design Development," *Integration, the VLSI Journa,* vol. 38, no. 2, pp. 131-183, 2004.

[88] A. Abraham and L. Jain, "Evolutionary Multiobjective Optimization," *Evolutionary Multiobjective Optimization,* pp. 1-6, 2005.

[89] J. L. Cohon, "Multicriteria Programming: Brief Review and Application," G. J. S, Ed., New York, Academic Press, 1985.

[90] A. S. M. Masud and C. L. Hwang, "Multiple Objective Decision Making – Methods and Applications," in *Lecture Notes in Economics and Mathematical Systems*, vol. 164, Springer, 1979.

[91] K. M. Miettinen, Nonlinear Multi-Objective Optimization, Boston: luwer Academic Publishers, 1999.

[92] L. M. Rios and N. V. Sahinidis, "Derivative-Free Optimization: A Review of Algorithms and Comparison of Software Implementations," *Advances in Optimization II,* 2009.

[93] S. P. Bradley, A. C. Hax and T. L. Magnant, Applied Mathematical Programming, Addison-Wesley, 1977.

[94] T. G. W. Epperly, Global Optimization of Nonconvex Nonlinear Programs Using Parallel Branch and Bound, Citeseer, 1995.

[95] P. N. Koch, J. P. Evans and D. Powell, "Interdigitation for Effective Design Space Exploration Using iSIGHT," *Structural and Multidisciplinary Optimization,* vol. 23, no. 2, p. 111–126, 2002.

[96] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall, 2009.

[97] "Hill climbing," [Online]. Available: http://en.wikipedia.org/wiki/Hill_climbing. [Accessed 24 October 2013].

[98] S. Kirkpatrick;, C. D. Gelatt and M. P. Vecchi, "Optimization by Simulated Annealing," *Science,* vol. 220, no. 4598, p. 671–680, 1983.

[99] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller and E. Teller, "Equation of State Calculations by Fast Computing Machines," *The Journal of Chemical Physics,* vol. 21, no. 6, p. 1087–1092, 1953.

[100] F. Glover, "Tabu Search—Part I," *ORSA Journal on Computing,* vol. 1, no. 2, p. 190–206, 1989.

[101] F. Glover, "Tabu Search—Part II," *ORSA Journal on Computing,* vol. 2, no. 1, pp. 4-32, 1989.

[102] E. Díaz, J. Tuya, R. Blanco and J. J. Dolado, "A Tabu Search Algorithm for Structural Software Testing," *Computers & Operations Research,* vol. 35, no. 10, p. 3052–3072, 2008.

[103] E. Diaz, J. Tuya and R. Blanco, "Automated Software Testing Using a Metaheuristic Technique Based on Tabu Search," in *Proceedings of the 18th IEEE International Conference on Automated Software Engineering*, 2003.

[104] A. Ravindran, K. M. Ragsdell and G. V. Reklaitis, Engineering Optimization Methods and Applications, Hoboken, NJ: John Wiley & Sons, 2006.

[105] D. E. Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning," *Machine Learning,* vol. 3, no. 2-3, pp. 95-99, 1989.

[106] L. Rela, Evolutionary Computing in Search-Based Software Engineering, Lappeenranta University of Technology, 2004.

[107] O. Räihä, Applying Genetic Algorithms in Software Architecture Design, University of Tampere, 2008.

[108] J. J. Dolado, "A Validation of the Component-Based Method for Software Size Estimation," *IEEE Transactions on Software Engineering,* vol. 26, no. 10, p. 1006 –1021, 2000.

[109] J. J. Dolado, "On The Problem of the Software Cost Function," *Information and Software Technology,* vol. 43, no. 1, p. 61–72, 2001.

[110] S. Wappler, Automatic Generation of Object-Oriented Unit Tests Using Genetic Programming, Berlin: Universitätsbibliothek der Technischen Universität, 2007.

[111] S. Wappler and J. Wegener, "Evolutionary Unit Testing of Object-Oriented Software Using Strongly-Typed Genetic Programming," in *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, 2006.

[112] J. H. Holland, "Genetic Algorithm and the Optimal Allocation of Trials," *SIAM Journal on Computing, vol 2, June 1973,* pp. 88-105, 1973.

[113] Z. He, G. G. Yen and J. Zhang, "Fuzzy-Based Pareto Optimality for Many-Objective Evolutionary Algorithms," *IEEE Transactions of Evolutionary Computation,* vol. PP, no. 99, pp. 1-16, 2013.

[114] Z. Michalewicz and M. Schoenauer, "Evolutionary Algorithms for Constrained Parameter Optimization Problems," *Evolutionary Computation,* vol. 4, no. 1, pp. 1-32, 1996.

[115] P. J. Fleming and C. M. Fonseca, "Multi-objective Optimization and Multiple Constraint Handling with Evolutionary Algorithms-Part I," *IEEE Transactions on Systems, Man, and Cybernetics- Part A: Systems and Humans,* vol. 28, no. 1, pp. 26-37, 1998.

[116] T. P. Runarsson and X. Yao, "Stochastic Ranking for Constrained Evolutionary Optimization," *IEEE Transactions on Evolutionary Computation,* vol. 4, no. 3, pp. 284-294, 2000.

[117] R. Farmani and J. A. Wright, "Self-Adaptive Fitness Formulation for Constrained Optimization," *IEEE Transactions on Evolutionary Computation,* vol. 7, no. 5, pp. 445-455, 2003.

[118] Y. G. Woldesenbet, G. G. Yen and B. G. Tessema, "Constraint Handling in Multiobjective Evolutionary Optimization," *IEEE Transactions on Evolutionary Computation,* vol. 13, no. 3, pp. 514-525, 2009.

[119] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer Methods Applied Mechanics and Engineering,* no. 186, pp. 311-338, 2000.

[120] C. A. Coello and E. Mezura-monte, "A Simple Multimembered Evolution Strategy to Solve Constrained Optimization Problems," *IEEE Transactions on Evolutionary Computation,* vol. 9, no. 1, pp. 1-17, 2005.

[121] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan and Q. Zhang, "Multiobjective Evolutionary Algorithms: A Survey of the State of the Art," *Swarm and Evolutionary Computation,* no. 1, pp. 32-49, 2011.

[122] Y. Wang, Z. Cai, G. Guo and Y. Zhou, "Multiobjective Optimization and Hybrid Evolutionary Algorithm to Solve Constrained Optimization Problems," *IEEE Transactions Systems, Man, and Cybernetics- Part B: Cybernetics,* vol. 34, no. 3, pp. 560-575, 200.

[123] Z. Cai and Y. Wang, "Combining Multiobjective Optimization with Differential Evolution to Solve Constrained Optimization Problems," *IEEE Transactions on Evolutionary Computation,* vol. 1, no. 16, pp. 117-134, 2012.

[124] W. A. Crossley and D. H. Laananen, "Conceptual Design of Helicopters via Genetic Algorithm," *Journal of Aircraft,* vol. 33, no. November-December, pp. 1062-1070, 1996.

[125] J. P. Gonzalez-Zugasti, K. N. Otto and J. D. Baker, "A Method for Architecting Product Platforms," *Research in Engineering Design,* no. 12, pp. 61-72, 2000.

[126] R. Hassan, O. de Weck and P. Springmann, "Architecting a Communication Satellite Product Line,"

in *22nd International Communication Satellite Systems Conference*, Monterey, CA, 2004.

[127] T. W. Simpson and B. S. D'souza, "Assessing Variable Levels of Platform Commonality within a Product Family Using a Multiobjective Genetic Algorithm," in *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Atlanta, GA, 2002.

[128] T.-L. Yu, A. A. Yassine and D. E. Goldberg, "An information theoretic method for developing modular architectures using genetic algorithms," *Research in Engineering Design,* vol. 18, no. 2, pp. 91-109, 2007.

[129] R. Hassan and W. Crossley, "Discrete Design Optimization under Uncertainty: A Generalized Population-Based Sampling Genetic Algorithm," in *AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics & Materials Conference*, Palm Springs, 2004.

[130] C. H. Dagli and R. Wang, "Developing a holistic modeling approach for search-based system architecting," *Procedia Computer Science,* vol. 16, pp. 206-215, 2013.

[131] C. H. Dagli, A. Singh, J. Dauby and R. Wang, "Smart Systems Architecting: Computational Intelligence Applied to Trade Space Exploration and System Design," *Systems Research Forum,* vol. 3, no. 2, p. 101–120, 2009.

[132] P. Brucker, A. Drexl, R. Möhring, K. Neumann and E. Pesch, "Resource Constrained Project Scheduling: Notation, Classification, Models, and Methods," *European Journal of Operational Research,* vol. 112, no. 1, pp. 2-41, 1999.

[133] F. B. Talbot, "Resource-constrained project scheduling with time-resource tradeoffs: The nonpreemptive case," *Management Science,* vol. 28, no. 10, pp. 1197-1210, 1982.

[134] W. Herroelen, B. De Reyck and E. Demeulemeester, "Resource-constrained project scheduling: a survey of recent developments," *Computers & Operations Research,* vol. 25, no. 4, pp. 279-302, 1998.

[135] M. B. Wall, "A genetic algorithm for resource-constrained scheduling," Doctoral dissertation, Massachusetts Institute of Technology, Cambridge, MA, 1996.

[136] K. Bouleimen and H. Lecocq, "A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version," *European Journal of Operational Research,* vol. 149, no. 2, pp. 268-281, 2003.

[137] A. Mostafavi, D. M. Abraham, S. Noureldin, G. Pankow, J. Novak, R. Walker, K. Hall and B. George, ""Risk-based Protocol for the Inspection of Transportation Construction Projects undertaken by State Departments of Transportation," *Journal of Construction Engineering & Management, ASCE,* vol. 139, no. 8, pp. 977-986, 2012.

[138] E. S. Maskin, "Theories of the Soft Budget Constraint," *Japan & the World Economy,* vol. 8, pp. 125-133, 1996.

[139] G. Dudek, M. R. M. Jenkin, E. Milios and D. Wilkes, "A taxonomy for multi-agent robotics," *Autonomous Robots,* vol. 3, no. 4, pp. 375-397, 1996.

[140] M. Amberg, "Modeling Adaptive Workflows in Distributed Environments," in *Proc. of the 1st Int. Conf. on Practical Aspects of Knowledge Management*, Basel, 1996.

[141] S. Ross, Introduction to Probability Models, 8th Edition, Academic Press, 2003.

[142] A. Shtub, J. F. Bard and S. Globerson, Project Management, New Jersey: Prentice Hall, 1994.

[143] B. Hunt, R. L. Lipsman and J. M. Rosenberg, A guide to MATLAB: For beginners and experineced users, New York: Cambridge University Press, 2001.

[144] F. Lopes, M. Wooldridge and A. C. Novais, "Negotiation among autonomous computational agents: principles, analysis and challenges," *Artificial Intelligence Review 29,* pp. 1-44, 2008.

[145] M. Bac and H. Raff, "Issue-by-issue negotiations: the role of information and time preference," *Games and Economic Behavior,* pp. 125-134, 1996.

[146] P. Faratin, C. Sierra and N. R. Jennings, "Negotiation decision functions for autonomous agents," *Robotics and Autonomous Systems 24,* pp. 159-182, 1998.

[147] B. Min and S. H. Chang, "System Complexity Measure in the Aspect of Operational Difficulty," *IEEE Transactions Nuclear Science,* vol. 38, no. 5, pp. 1035-1039, 1991.

[148] J. A. C. III, "A Quantitative approach to Assessing System Evolvability," *Systems Engineering,* vol. 770, pp. 1-16, 2004.

[149] S. Agarwal and C. H. Dagli, "Augmented Cognition in Human–System Interaction through Coupled Action of Body Sensor Network and Agent Based Modeling," in *Conference on System Engineering Research*, 2013.

[150] Department of the Navy, "Contractor Performance Assessment Reporting System (CPARS)," Washington DC, 1997.

[151] D. Dombkins, Complex Project Management, South Carolina: Booksurge Publishing, 2007.

[152] D. S. Alberts, J. J. Garstka and F. P. Stein, Network Centric Warfare: Developing and Leveraging Information Superiority, 2nd Edition, Washington DC: C4ISR Cooperative Research Program, 1999.

[153] P. K. Dutta, Strategies and Games Theory & Practice, The MIT Press, 1999.

[154] B. W. Lamar, "Min-additive utility functions," MITRE Corporation, 2009.

[155] D. H. Dombkins, "Project Managed Change: The Application of Project Management Techniques to Strategic Change Programs," University of New South Wales, Sydney, 1996.

[156] M. Thompson, "Iraq: The Great Scud Hunt," *Time Magazine,* pp. http://www.time.com/time/magazine/article/0,9171,1003916,00.html, 23 December 2002.

[157] W. Rosenau, "Coalition Scud Hunting in Iraq, 1991," RAND Corporation, 1991.

[158] L. Pape and C. Dagli, "Assessing robustness in systems of systems meta-architectures," in *Complex Adaptive Systems*, Baltimore, 2013.

[159] G. Contag, C. Laing, J. Pabon, E. Rosenberg, K. Tomasino and J. Tonello, "Nighthawk System Search and Rescue (SAR) Unmanned Vehicle (UV) System Development," SE4150 Design Project, Naval Postgraduate School, 2013.

[160] W. Johnston, K. Mastran, N. Quijano and M. Stevens, "Unmanned Vehicle Search and Rescue Initiative," SE4150 Design Project, Naval Postgraduate School, 2013.

[161] A. Gegov, Fuzzy Networks for Complex Systems, Berlin: Springer, 2010.